IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 October 2023, accepted 1 November 2023, date of publication 13 November 2023,
date of current version 22 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3332483

RESEARCH ARTICLE

A Novel Approach of Latency and Energy Efficiency Analysis of IIoT With SQL and NoSQL Databases Communication

SOUKAINA BAKHAT KENITAR[1], MOUNIR ARIOUA[1], AND MOHAMMED YAHYAOUI[2]
[1]Laboratoire Ingénierie des Systèmes, Innovants School: National School of Applied Sciences, Abdelmalek Essaadi University, Tetouan 93000, Morocco
[2]Faculty of Science, Technology and Modeling Systems Research Unit, Abdelmalek Essaadi University, Tetouan 93000, Morocco

Corresponding author: Soukaina Bakhat Kenitar (soukaina.bakhat@uae.ac.ma)

ABSTRACT Industrial Internet of Things (IIoT)-enabled production facilities generate vast amounts of data, which, if harnessed effectively, can substantially enhance manufacturing efficiency through latency reduction. The selection of the appropriate data storage technology is a pivotal consideration in achieving this objective. While prior studies have examined SQL and NoSQL databases in terms of latency and energy efficiency, these evaluations have not been conducted specifically within the context of IIoT. This paper aims to fill this research gap by conducting a rigorous comparison of SQL and NoSQL databases, focusing on their performance latency and energy efficiency when interfaced with IoT nodes. By elucidating these relationships, our research offers actionable insights that can guide IIoT-enabled manufacturing facilities in optimizing their operations. Specifically, the paper aids in the selection of the most suitable database technology, thereby contributing to latency minimization and efficiency maximization in industrial settings.

INDEX TERMS SQL, NoSQL, latency, efficiency, IIoT, energy.

I. INTRODUCTION

The performance analysis and comparison of SQL and NoSQL databases have been a vibrant field of research lately [1]. Each of these databases has unique advantages and disadvantages. These comparisons have been studied from various points of view focusing on different types of industries [2]. However, it has been observed that the performance of IoT nodes' communication with SQL and NoSQL databases in terms of latency and energy efficiency is an under-explored area [3]. This paper presents an exploratory analysis to study and analyze the effect of NoSQL and SQL databases on the latency and energy efficiency of IoT. The findings of this exploratory analysis are a valuable insight that has the potential to improve the efficiency of the Industrial Internet of Things (IIoT) by guiding the manufacturing industries to choose the most appropriate solution for their IoT data storage challenges.

This is the first-ever exploratory analysis that explores this field from both latency and energy efficiency perspectives. Initially, this paper presents an informative view of NoSQL and MySQL databases and protocols that govern IoT communication with the databases. After that, a unique methodology has been developed to conduct the analysis. It starts with the data description and processing. After that, the data schema has been studied. The performance of the IoT nodes in terms of latency and energy efficiency for NoSQL and SQL databases has been studied through load testing which have been detailed in this paper as well. The experimental setup design is one of the innovative aspects of this paper which makes it unique from other similar studies. The core novelties and unique contributions of this paper are listed below:

1) **Novel Efficiency Modeling:** A creative and efficiency model formation to measure the energy efficiency of SQL and NoSQL IIoT communication for the first time ever.
2) **Unique Exploratory Analysis:** Performance analysis of NoSQL and SQL databases for IoT node communication in terms of latency and energy efficiency through simulation of an industrial environment.

The associate editor coordinating the review of this manuscript and approving it for publication was Adamu Murtala Zungeru.

3) **Innovative Experimental Setup:** An innovative and unique experimental setup has been designed and implemented in this study.
4) **Impactful Findings:** The findings of this paper are the first discovery of differences between latency for different number production units consisting of IoT nodes for both SQL and NoSQL.

The remaining part of the paper has been organized into six different sections. The literature review has been presented in the second section where the novelty of the proposed paper has been compared with the research gap available in other papers. The database and related protocol for the analysis have been studied in the third section. The fourth section discusses the methodology of this study. The fifth section presents the experimental result and analysis. The limitations and future scope of this paper have been discussed in the sixth section. Finally, the paper has been concluded in the seventh section.

II. LITERATURE REVIEW

P. S. Mehta [4] investigates the pros and cons of integrating NoSQL databases like MongoDB and Cassandra into Kubernetes (K8s). The K8s Operator setup and YCSB benchmark performance assessment are the emphasis. The project promotes K8s-integrated databases, guides their deployment, and provides performance insights. Instead of traditional databases, NoSQL databases enable dynamic schema modifications during runtime, raising doubts regarding the schema's importance. However, this study does not address the latency and efficiency which have been studied in this paper. Bansal et al. [5] examines Document, Column, and Key-Value NoSQL databases. An example Entity Relationship (ER) model tests schema architectures for query performance, including response time, speed, and database size. The results show that schema decisions affect NoSQL database performance. This paper is a critical analysis from a schema architecture perspective. While it is useful within the context of the paper, the application of this study is confined to the schema architecture domain. The proposed approach is robust and discovers valuable decision-making criteria for industrial production efficiency improvement. da Silva et al. [6] test Docker Swarm's cost-effectiveness for cloud service scalability and high availability. The results balance performance and replication, which are essential in low-power distributed systems. Citus and HBase's consistency hindered performance as replication grew, while Cassandra fared well with customizable consistency. This promising analysis finds the optimal balance for a low-power distribution system. However, it does not demonstrate the latency and efficiency of the optimized approach which has been studied in the proposed approach.

Andreoli et al. [7] tweaks MongoDB NoSQL to allow OS-level priority-based performance modifications. The change is modest, doesn't affect database functionality, and interacts with MongoDB's security access control. This method prioritizes response times for high-priority customers in mixed-client settings, according to tests. It is a study of SQL and NoSQL performance from a customer's priority perspective. Based on the priority, this approach switches between SQL and NoSQL. It does not effectively compare the two approaches from common ground the way the proposed approach has done. Gomes et al. [8] use stochastic Petri nets to assess cloud-based NoSQL database consistency. The models evaluate system speed, availability, and data freshness. The study also analyzes consistency-related energy usage. The approach is viable based on experiments. This paper analyzes important aspects of NoSQL and discovers different characteristics. However, it is a unipolar study without side-by-side analysis and comparison of SQL and NoSQL approaches which has been done in the proposed methodology. Arshad et al. [9] discuss NoSQL databases in Big Data Analytics, including their characteristics, kinds, and commercial advantages. Due to massive volumes of unstructured data, data management paradigms must change. This study compares RDBMS with NoSQL databases, stressing NoSQL's applicability for unstructured Big Data analytics. The concept of this paper aligns with the proposed approach. However, this paper has extended the study up to latency and energy efficiency analysis which has never been done before [10].

Khan et al. [11] compare Oracle RDBMS with MongoDB on cloud data migration and SQL and NoSQL database software architectures. NoSQL databases excel in large data analytics, whereas SQL databases excel at transaction processing. This comparison helps to craft efficient storage architecture with appropriate DBMS. However, this study only focuses on database migration and ignores valuable performance measurement metrics such as latency and energy consumption. The proposed methodology abridges the gap of this paper by evaluating the performance using these two metrics. Khan et al. [12] analyze and organize 80 NoSQL technologies for easier comprehension. These solutions are classified, compared, and used on huge graphs in six categories. They also have a decision tree model and online application to help people choose NoSQL. It is a unique study and reveals valuable features of NoSQL database applications. However, this paper leaves the scope of raising a question about the effect of SQL databases on the same settings. The proposed analysis leaves no such scope. It is a comprehensive study and the first of its kind study that compares SQL and NoSQL from a latency and energy efficiency perspective. Kim et al. [13] tested MongoDB and Couchbase, two popular document storage, and offered findings and observations. A complicated dataset was used to test Apache Accumulo with the GeoMesa framework to demonstrate GeoYCSB's versatility. It is another study that compares two NoSQL approaches. This analysis is from a utility perspective and studies two different types of NoSQL databases. The proposed study is robust, involves both SQL and NoSQL comparison, proposes unique ways of evaluating performances, and demonstrates an innovative methodology

to analyze the performance of latency and energy efficiency for IIoT communication with SQL and NoSQL database using MQTT protocol.

III. DATABASE AND PROTOCOL ANALYSIS

A. AWS AURORA SQL DATABASE

SQL requires a predefined data field to store data. Otherwise, there is no exception to store information. Because of this nature, the SQL database schema is defined as a rigid schema. It supports both single server operation and clustered server configurations, illustrated in figure 1. Furthermore, it supports atomic transactions where unity information can be retrieved from or inserted into the table. These characteristics have made it a reliable data storage system for enterprises. However, the rigid nature of the schema and field-specific query develops complex relationships among multiple fields as the database grows larger [14].

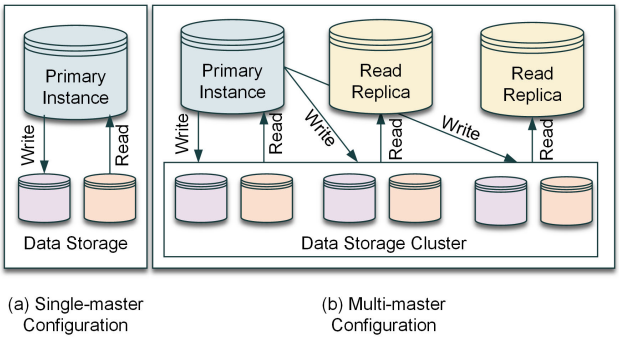

FIGURE 1. AWS Aurora single-master and multi-master configuration.

This paper uses the AWS relational database, Aurora, which comes in single and multi-master configurations. The single-master configuration is vertically scalable, and the system performance depends on the availability of the hardware. On the other hand, the multi-master configuration is horizontally scalable, enabled by clustering among multiple servers. The experimental context is to evaluate the latency and energy consumption which become challenging to track for multi-master configuration. That is why the single-master configuration has been used in this paper.

B. AWS DYNAMODB NOSQL DATABASE

The NoSQL databases do not have rigidly defined table schema. Instead of the key-value pair and key-based document stores are used where these keys are used to retrieve the data. Unlike SQL databases, predefined fields are not mandatory for NoSQL approaches. However, data are comparatively less normalized in NoSQL databases. As a result, data duplication is a common issue in this method. The absence of complex relations among multiple fields makes NoSQL simpler. However, retrieving stored records requires advanced knowledge of the query pattern. These exclusive characteristics enable faster data writing and retrieval process [15].

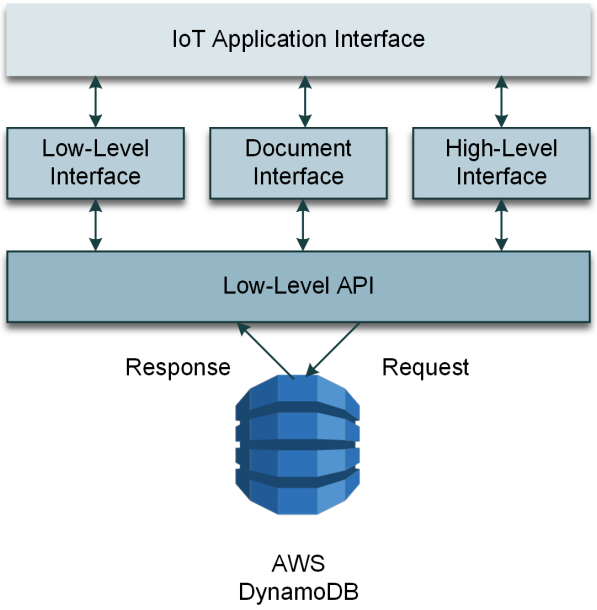

FIGURE 2. AWS DynamoDB NoSQL database communication architecture.

This paper uses the AWS DynanoDB NoSQL database illustrated in figure 2. It has been used in this paper following the AWS service model for DynamoDB. The Application Interface communicates with the experimental IoT through low-level, document, and high-level interfaces. The low-level API sits in between these layers and the database. Because of being a NoSQL approach, this communication does not require any field specification.

C. PROTOCOL ANALYSIS

The IoT as a Service (IoTaaS) offers zero-code integration of IoT nodes with both SQL and NoSQL databases [16]. It uses the Message Queuing Telemetry Transport (MQTT) protocol for this communication [17]. The application of MQTT protocol for IoT communication has been observed in healthcare [18], robotics [19], cyber-physical system security [20], smart home [21], industrial production [22], and in many different sectors. It is a widely used lightweight Publish-Subscribe (Pub/Sub)-based protocol popular for efficiency and easier implementation on various devices. Let C be the set of all clients, and T be the set of all topics. Then the function that represents the subscription of a client to a topic is defined by equation 1 where if $S(c, t) = 1$, client c is subscribed to topic t. Otherwise, if $S(c, t) = 0$, client c is not subscribed to topic t.

$$S : C \times T \rightarrow \{0, 1\} \tag{1}$$

The function representing publishing a message to a topic is expressed by equation 2. When $P(c, t, m)$ is called, the message m is sent to all clients c' such that $S(c', t) = 1$.

$$P : C \times T \times M \rightarrow M \tag{2}$$

IV. METHODOLOGY

A. DATA DESCRIPTION & PROCESSING

The data has been collected from an automated branch of a manufacturing firm. The production floor of the industry has been running for thirteen years. And the automated branch has been in production since 2019. Each automated unit has multiple sensors that communicate over the cloud server through connected IoT devices. The sensor signals and corresponding responses are stored in a MySQL database. The data have been collected from this database with the permission of the respective authority. The IoT data are stored in the MySQL database using the MQTT protocol.

1) OUTLIER ELIMINATION

The MQTT payload size stored in the database ranges between 1×10^2 to 1×10^3 bytes [23]. The mean and standard deviation of the instances of each unit have been measured using equation 3 and 4 [24].

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3}$$

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{4}$$

The mean, $\bar{x}$, of a dataset is computed by summing all the values and then dividing by the number of values, n. It gives the average value of the dataset. The standard deviation, s, measures the amount of variation or dispersion from the mean. It's calculated by taking the square root of the average of the squared differences from the mean. The standard deviation gives insight into how spread out the values are around the mean. The mean and standard deviation are used to identify possible outliers in the dataset [25].

2) DATASET SCHEMA

The dataset collected from the industry has been retrieved from the organizational database and stored in the experimental database along with the metadata. The fields, their datatypes, and descriptions as comments have been presented in table 1. The dataset schema has been prepared to make it effective in calculating latency and analyzing energy efficiency.

TABLE 1. The dataset schema of the experimenting data storage.

Column	Type	Comment
Id	int(8)	Auto increment unique identifier
ReceivedDateTime	timestamp	UNIX timestamp
topic	varchar	CSV MQTT topic
payload	varchar	JSON object
assetId	varchar	Asset Unique Identified
energyConsumption	varchar	Consumed energy

B. LOAD TESTING STRATEGIES

The Apache JMeter 5.6.1 has been used in this experiment as the load testing tool [26]. It has been integrated into the AWS cloud server to analyze and measure the performance of the IoT nodes from evaluation metrics [10]. The Apache JMeter supports communication over the MQTT protocol which is one of the main reasons behind choosing this tool [27]. It has an open-source plugin that has interfaced with the IoT node communication. In this experiment, the client has been simulated using the JMeter. It transmits messages from AWS Elastic Cloud (EC2) instance [28]. These messages are received by the AWS IoT core service end-point. In this communication, each sensor has been considered as a single client. The unique asset ID has been used to uniquely define each client and track the communication.

1) TEST PLAN

The plan has been designed with three stages. The first stage is connecting to the client using the MQTT protocol. The second stage is about looping the messages through a logic controller. And the last stage is the MQTT disconnection stage. During the process of developing the test plan, we made use of an aggregate report listener in addition to a summary report listener in order to gather findings after the thread group. A constant throughput timer was utilized inside the message loop logic controller in order to produce traffic. The timer was programmed using parameters taken from the data set in order to get the desired results. A Gaussian timer was added to the model in order to generate noise, and its configuration was determined by taking the parameters from the data set [29]. The test plan has been structured in a user group. Each group operates through the three stages mentioned earlier.

a: COMMAND LINE PARAMETERS

Several command line parameters have been used to control the test configurations. These parameters are flags, clients, and duration. Every test plan is performed within a single user group. The user group configuration has been listed in table 2. Each group contains the MQTT connect module which is executed first. After connecting, the program enters into the message loop. Unless interrupted, the loop keeps running as long as there are messages from the IoT nodes. This loop is controlled through the configurations and corresponding values provided in table 2.

TABLE 2. The user group configuration and properties.

Configuration	Primary Value	Input Type
Number of users	1	Text
Connection Time	0	Number
Number of loops	1	Number
Loop Count	FALSE	Boolean
User switching	TRUE	Boolean
Thread creation delay	FALSE	Boolean
Thread lifetime	FALSE	Boolean
Energy consumption	TRUE	Boolean

b: EXECUTION LOOP

The main execution loop is controlled by the runtime controller. It controls the execution of the MQTT messages

within the defined timeline specified by the command line. During this loop, two functionalities are carried out. And they are the transmission of MQTT messages and controlling the timing. As long as the thread lifetime is not specified, the loop continues. In this loop, the MQTT publisher transmits the MQTT messages to a predefined topic for every client. This communication carries a sequence of characters in JSON format.

Every time the MQTT publisher transmits a message, the time() method is initiated [30]. The duration between the transmission of the first and last bit is tracked by this method. Finally, the timestamp is embedded in the message itself which is retrievable. It has been observed that there are some random delays associated with the timestamp in real communication. Initially, it was absent in the testing phase. However, later a Gaussian random timer has been configured and applied in the transmission process. After this modification, the differences between actual and testbed timestamps were marginal.

c: ENERGY CONSUMPTION

The purpose of tracking energy consumption data is to measure the efficiency of IoT devices. The more messages IoT devices transmit, the more energy they consume. However, it has a direct relation with the MySQL and NoSQL databases. These two databases handle messages differently. As a result, the energy consumption is also different. Through an energy consumption test, this paper finds out the efficiency of the IoT nodes for MySQL and NoSQL databases [31].

d: LATENCY MEASUREMENT

The AWS Application Programming Interface (API) offers scope to retrieve the latency of the communication over the MQTT protocol for both SQL and NoSQL databases [32]. The process starts by recording the time of the initial state and the terminal state of the execution loop. There is a time difference between the loop execution and the time the messages are retrieved from the database. This time difference is an effective measure of the latency.

C. THE EXPERIMENTAL SETUP

The experimental setup is different from the production environment from where the data have been collected. However, workflow is logically equivalent to the actual environment.

1) ACTUAL ENVIRONMENT

The production facility has multiple robotic arms that take care of the production. Each arm contains multiple sensors. These sensors sense the environment and according to the intelligence in the embedded system, it act upon the environment [33]. The data created during this sensing and execution are sent to the MySQL database using the MQTT protocol. Regardless of the architectural complexities, the communication method of the entire system has been simplified as illustrated in figure 3.

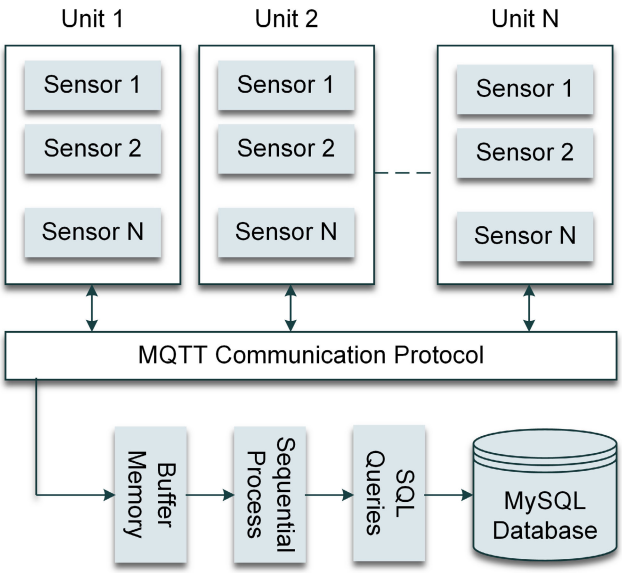

FIGURE 3. The actual communication model of the experimenting IoT network.

There is a buffer memory to hold the data temporarily. The data in the buffer memory are processed through the Sequential Processor and generate SQL queries for each message [34]. The signals from the sensors carry unique asset IDs. According to the nature of sensors, there are predefined tables created in the MySQL database. The sequential processor orders the messages for SQL queries [35]. A layer responsible for SQL queries performs the queries and stores the data in the MySQL database.

2) IOT SIGNAL PROCESSING

Every unit of figure 3 represents a single robotic arm that contains multiple sensors. Mathematically the sensors are organized as defined in equation 5 where S_N represents N^{th} sensor. The sensor data are formatted as 'key-value paid' defined as equation 6. Here, *uid*, *sid*, *d*, *sts*, *ets*, and *erg* refer to Unit ID, Sensor ID, Data, Starting Timestamp, Ending Timestamp, and Energy, respectively. The key-value paired data are generated in the data layer of MQTT protocol.

$$\begin{aligned} S_{array} = \{&(S_1, S_2, \ldots, S_N),\\ &(S_1, S_2, \ldots, S_N),\\ &\ldots,\\ &(S_1, S_2, \ldots, S_N)\} \end{aligned} \tag{5}$$

$$\begin{aligned} M_{data} = \{&\text{“uid”: “16004”},\\ &\text{“sid”: “0003”},\\ &\text{“d”: “11011001”},\\ &\text{“sts”: “1692170173”},\\ &\text{“ets”: “1692170175”},\\ &\text{“erg”: “0.346Wh”}\} \end{aligned} \tag{6}$$

The data generated in the data layer are converted into separated strings for every data instance. The MQTT topic is attached to this string in this layer. The core operations of the application layer are defined in equation 7 where the $Str()$ function converts M_{data} into a string and the $TPC()$ function adds the topic. The $f_{app}()$ is the application layer function which adds the layer-specific header for the subsequent layers. And M_2 is the final output from this layer.

$$M_{app} = f_{app}(Tpc(Str(M_1))) \tag{7}$$

The transport layer encapsulates M_{app} inside a TCP segment [36]. It also encrypts the data and generates data for the network layer which is defined in equation 8. The network layer encapsulates the M_{tcp} within IP packets which are expressed by equation 9 where M_{net} represents the IP packet. Finally, the M_{net} is converted into an Ethernet frame which is expressed by equation 10.

$$M_{tcp} = f_{tcp}(M_{app}) \tag{8}$$
$$M_{net} = f_{net}(M_{tcp}) \tag{9}$$
$$M_{lnk} = f_{lnk}(M_{net}) \tag{10}$$

Finally, the link layer carries the data to the buffer memory. The sequential processor after the buffer memory processes the data packets one by one. Each packet represents one instance of data generated by an IoT sensor of a production unit. Predefined queries are performed to store these data in the MySQL database. This is how the IoT signals are processed in the actual production environment [37].

3) SIMULATED ENVIRONMENT

A simulated environment has been created to conduct the experiment which is illustrated in figure 4. The simulated environment replicates the characteristics of the actual environment illustrated in figure 3 and generates identical responses. Although the underlying technology is different, these two systems are logically equivalent according to their final responses. The sensor data are stored in a MySQL database in the actual production environment. Figure 4a is that MySQL database. SQL queries followed by a parallel processing module retrieve the sensor data and create the data stream which replicates the actual sensor data stream which is presented in figure 4b. Finally, there is a decision module that is used to select between MySQL and NoSQL databases which are illustrated in figure 4c.

V. EXPERIMENTAL RESULT AND ANALYSIS

A. LATENCY ANALYSIS

There is a time delay between the system responses and the storage time of those responses. This delay is the latency in the overall communication. One of the purposes of this study is to analyze the difference between the latency of MySQL and NoSQL databases for IoT data.

1) LATENCY IN MYSQL DATABASE

The average latency for 1 unit over 60 seconds period has been illustrated in figure 5. The horizontal axis shows the time duration in seconds and the vertical axis is for average latency in milliseconds. The average latency for the first experiment is 228.5 ms. The time duration has been increased by five seconds starting from five seconds to sixty seconds. It has been observed that the standard deviation is 23.42 ms which represents marginal differences among the latency for different time duration. However, it is noticeable in figure 5 that the latency increases when the time duration increases.

The latency for 10 units is different than the linearly incremental characteristics of latency for 1 unit. It has been illustrated in figure 6. Although the trends curve in figure 6 shows incremental characteristics for several instances, there is no observable pattern. It clearly indicates that the latency does not maintain any pattern with the number of units. However, the average latency for 10 units is 335.92 ms and the standard deviation is 28.22 ms. It clearly indicates that even if there is no traceable pattern, if the number of units increases, the latency increases.

The irregularity still exists after increasing the number of units to 100 which has been illustrated in figure 7. The characteristics curve demonstrates nature similar to the 10-unit experiment. However, the average latency is 824.75 ms for 100 units simultaneously. And the standard deviation is 87.48 ms.

2) LATENCY IN NOSQL DATABASE

The same experimental setup, time duration, and number of units have been used in the NoSQL database to compare the performances. It has been observed that the average latency for the NoSQL database is 199.67 ms and the standard deviation is 25.36 ms. The latency variations at different times have been illustrated in figure 8. It is clearly observable that the latency increases when the time duration increases. However, the rate of increase is not as smooth as it is for SQL databases.

The latency of the NoSQL database over 50 seconds with 10 units has been illustrated in figure 9. The average latency is 279.92 ms and the standard deviation is 27.057 ms. Although the average latency of NoSQL is lower than the MySQL database for IoT nodes, the characteristics curve is similar to the MySQL database. It does not have any traceable pattern.

The latency characteristics of the NoSQL database for IoT nodes with 100 units over 60 seconds have been illustrated in figure 10. The average latency is 679.75 ms and the standard deviation is 86.02 ms. This observation leads to the conclusion that the latency of NoSQL databases is lower than MySQL databases. However, the characteristics curve demonstrates a similar pattern for both MySQL and NoSQL.

B. ENERGY EFFICIENCY

Each sensor sends the energy consumption data. These data have been used to discover if there is any relation between

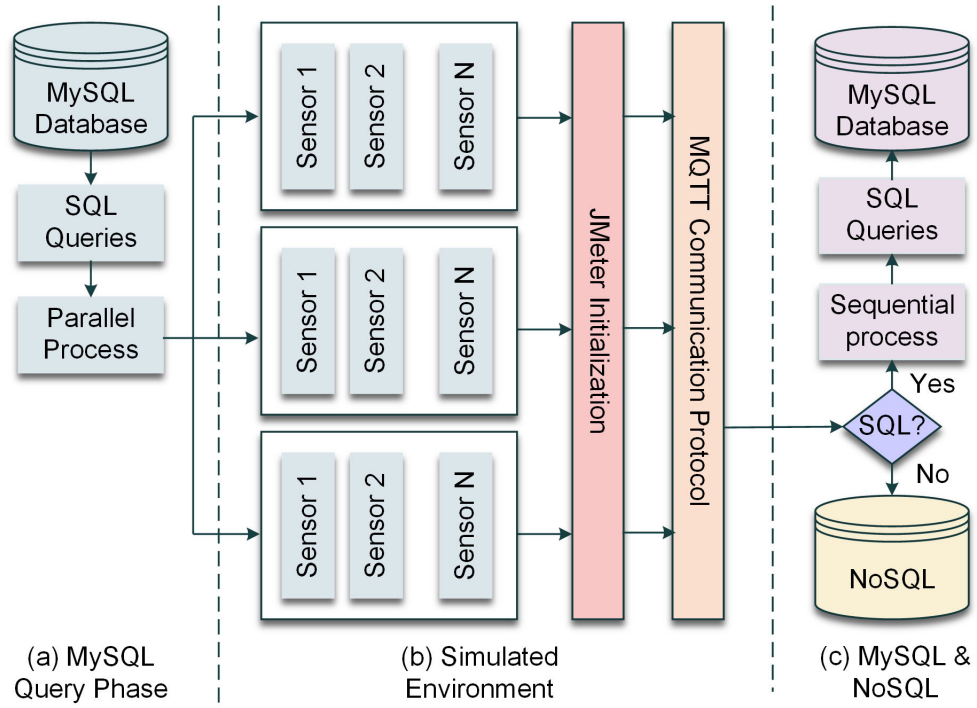

FIGURE 4. The simulated experimental environment.

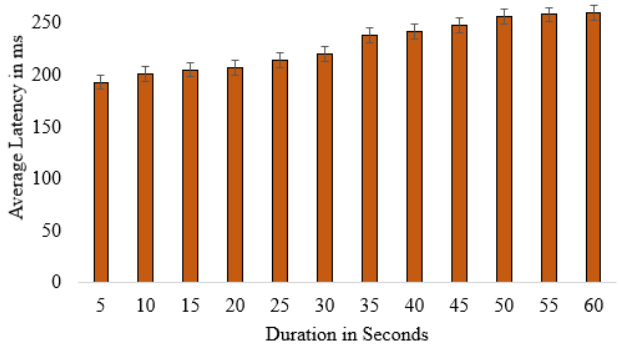

FIGURE 5. The average latency for 1 unit for MySQL database.

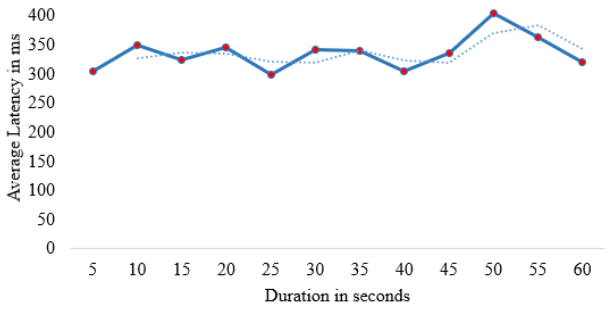

FIGURE 6. The latency variation for 10 units for MySQL database.

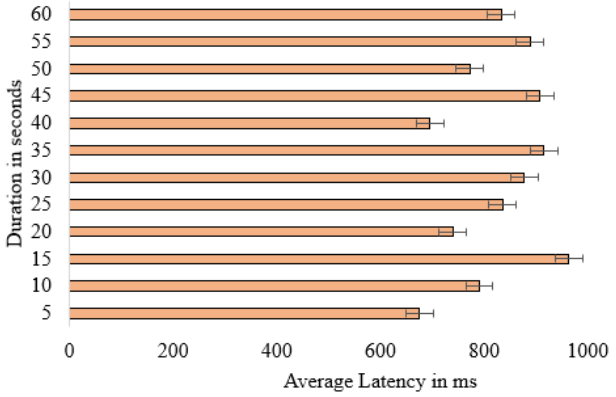

FIGURE 7. The latency variation for 100 units for MySQL database.

the energy efficiency of the production units with the type of database they are connected to. It has been observed that the energy efficiency of the production units does not show any significant variations when the database type is switched to NoSQL from MySQL.

1) ENERGY EFFICIENCY MODELING

The energy efficiency of a system is the ratio of useful output power and total input power expressed in a percentage calculated using equation 11. However, the experimental setup consisted of multiple elements consuming power at different rates. At the same time, the output of the system is measured in rates of data storage capability and system delay instead of power. This makes energy efficiency calculation challenging using the equation 11. A novel energy efficiency measurement model has been designed, implemented, and

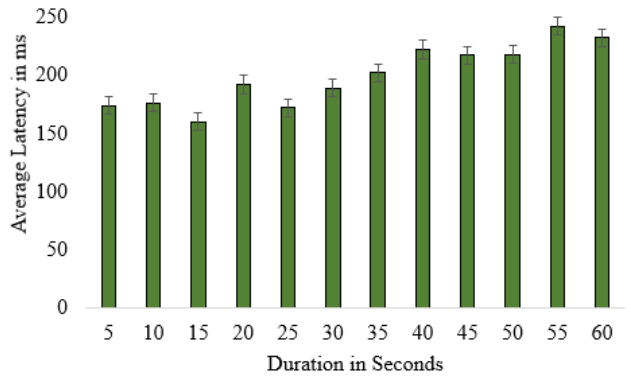

FIGURE 8. The average latency for 1 unit for NoSQL database.

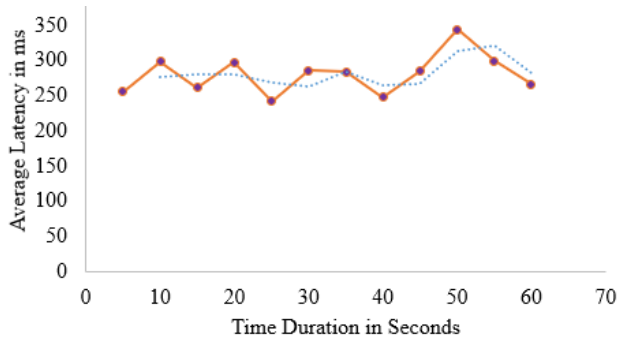

FIGURE 9. The average latency for 10 units for NoSQL database.

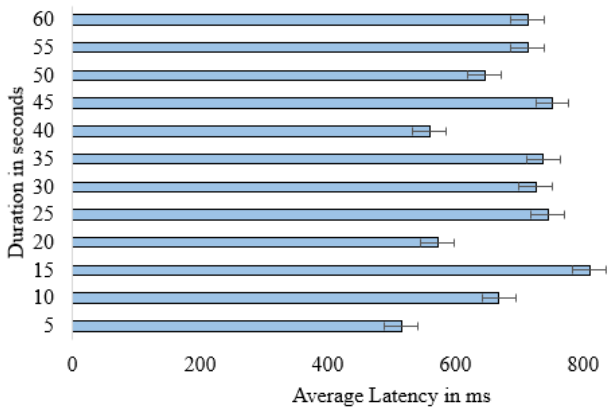

FIGURE 10. The average latency for 100 units for NoSQL database.

experimented in this experiment which is illustrated in figure 11.

$$E = \frac{P_{out}}{P_{in}} \times 100\% \tag{11}$$

The proposed energy efficiency model is presented in figure 11 calculates the Total Energy consumed by the IIoT Sensor (E_{IIoT}) units before passing the data through the MQTT protocol. The Total Number of Data Instances (T_{data}) is calculated during the network packet generation phase. The E_{IIoT} and T_{data} are the same for both SQL and NoSQL databases. It is beyond the scope of measuring the energy consumption at the electronics level by the DBMSs. However, this experiment has developed a novel alternative by running the SQL and NoSQL DBMSs in two different Virtual Machines (VMs).

2) ENERGY CONSUMPTION

The energy consumed by the SQL DBMS is characterized by Buffer Memory (B_m), Sequential Processing (S_p), SQL Query (S_q), and SQL DB Read/Write (DB_{srw}). These are the distinct units of the SQL DBMS running in an SQL VM. The distinct elements of the NoSQL DBMS running in the NoSQL VM are Parallel processing (P_p), Key-Value Access (K_a), and NoSQL DB Read/Write (DB_{rw}). Every consumption by the other Electronics Module (Et_m) of each VM has been considered logically equivalent in this experiment as the configurations of both VMs are the same. The union among all elements of Virtual Machine of SQL (VM_{SQL}) and Virtual Machine for NoSQL (VM_{NoSQL}) presented in equation 12 shows that the Et_m is common for both. The equation 13 further validates that the Et_m is common for both VMs.

$$(VM_{SQL} \cup VM_{NoSQL}) = \{B_m, S_p, S_q, DB_{srw}, P_p, K_a, DB_{rw}, Et_m\} \tag{12}$$

$$(VM_{SQL} \cap VM_{NoSQL}) = \{Et_m\} \tag{13}$$

From this observation, it is evident that the energy consumed by Et_m does not play an active role in overall efficiency calculation. The energy consumption by VM_{SQL} and VM_{NoSQL} are defined by equations 14 and 15, respectively.

$$E(VM_{SQL}) = E(B_m) + E(S_p) + E(S_q) + E(DB_{srw}) \tag{14}$$

$$E(VM_{NoSQL}) = E(P_p) + E(K_a) + E(DB_{rw}) \tag{15}$$

3) DATA R/W RATE

Both SQL and NoSQL databases Read and Write (R/W) all of the data received through the MQTT protocol. There is no data loss. However, the data R/W rates are different. The comparison between SQL and NoSQL reading and writing time for different numbers of data instances have been listed in table 3. The number of data instances are 64, 128, 256, and 512. For instance, at 64 instances, the writing time for SQL was 32.54 ms whereas for NoSQL it was slightly lower at 29.18 ms. The reading time followed a similar trend, with SQL taking 30.19 ms and NoSQL only 24.75 ms. As the number of instances doubles to 128, the writing time for SQL increased to 67.76 ms, while NoSQL lagged just slightly behind at 65.3 ms; for reading, SQL took 66.24 ms compared to NoSQL's 63.47 ms. At 256 instances, SQL had a writing time of 141.33 ms and a reading time of 135.45 ms, while NoSQL recorded 132.86 ms and 126.58 ms, respectively. For the highest tested instance count of 512, SQL registered a writing time of 285.71 ms and a reading time of 229.3 ms. NoSQL, on the other hand, required only 219.41 ms for writing and 201.34 ms for reading. Overall, while both SQL and NoSQL scale reasonably well as the

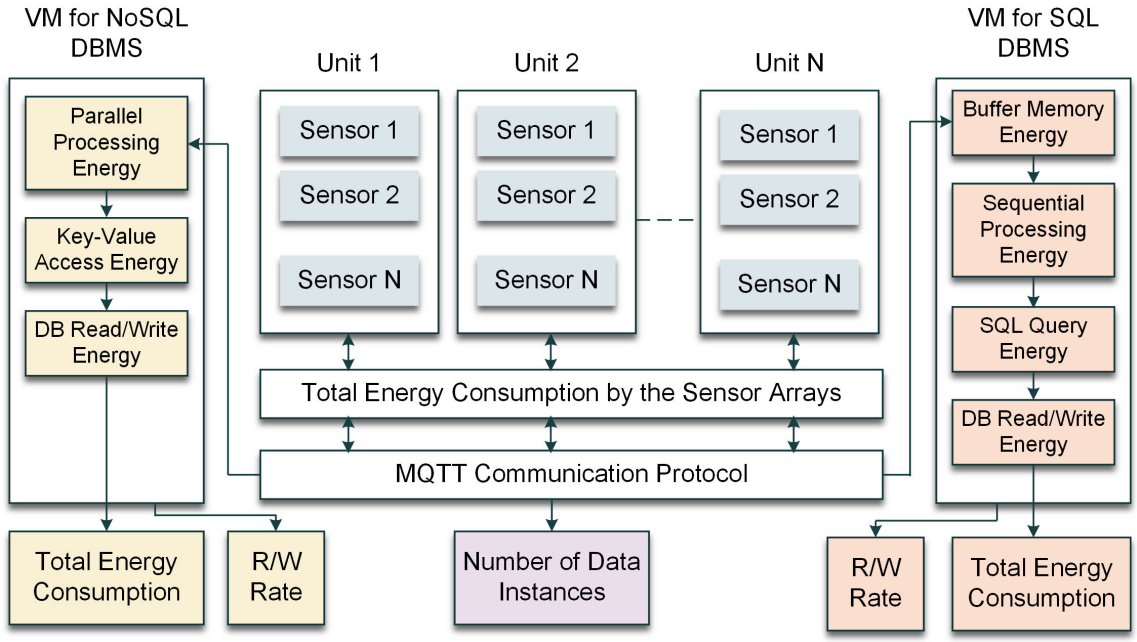

FIGURE 11. The model to measure and compare the energy efficiency of SQL and NoSQL database for IIoT data.

number of instances increases, NoSQL appears to offer a slight performance advantage, especially in terms of reading time. The comparison has been illustrated in figure 12 to visualize the differences.

TABLE 3. Read/Write response time for SQL and NoSQL database for different number of instances.

	SQL (ms)		NoSQL (ms)	
Instance Received	Writing Time	Reading Time	Writing Time	Reading Time
64	32.54	30.19	29.18	24.75
128	67.76	66.24	65.3	63.47
256	141.33	135.45	132.86	126.58
512	285.71	229.3	219.41	201.34

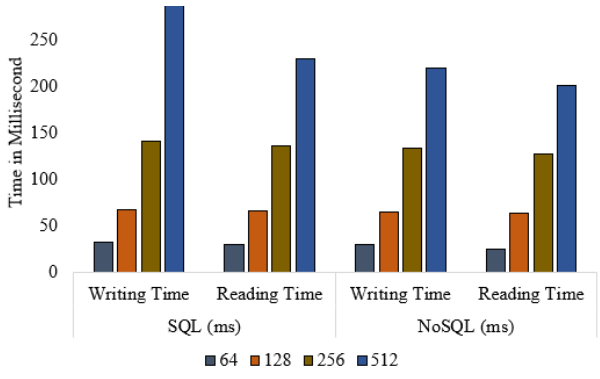

FIGURE 12. Comparison of reading and writing rate between SQL and NoSQL database.

The average reading and writing time for SQL and NoSQL have been calculated using equation 16 where $x = \{read, write\}$ and $j = \{SQL, NoSQL\}$. To write 240 data instances, the SQL takes 131.835ms and NoSQL takes 111.687ms. On the other hand, the reading times for the same number of data instances are 115.295ms and 104.035ms for SQL and NoSQL, respectively. The SQL DBMS takes 1.055s to read and write 1024 data instances. On the other hand, the NoSQL DBMS takes 0.92s for the same number of instances. According to equations 14 and 15, the energy consumed per second by VM_{SQL} and VM_{NoSQL} are 32.19 j/s and 28.01 j/s, respectively.

$$avg(x) = \frac{\sum_{i=1}^{i=4} x_i^j}{\sum_{k=1}^{k=4} k} \tag{16}$$

4) EFFICIENCY CALCULATION

The input to the system is the data instances. The output from the system is the reading and writing operations. To process 1024 input data instances, the SQL and NoSQL DBMSs take 33.96 j/s and 27.45 j/s energy, respectively. The number of data processing per second by SQL and NoSQL DBMS are calculated by equation 17. It concludes that the SQL DBMS receives 1024 instances per second and processes 971 instances per second. On the other hand, the NoSQL DBMS receives the same number of instances and processes all of them in 0.92 seconds. Therefore the efficiency of SQL DBMS is 94.81% and the efficiency of NoSQL DBMS is 98.00%.

$$D_{instance} = \frac{D_{avg} \times 1000}{\frac{\sum_{i=1}^{j=4}(W_i+R_i)}{\sum_{k=1}^{k=4} k}} \tag{17}$$

VI. LIMITATION AND FUTURE SCOPE

This paper discovers valuable and unique insights into the latency and energy efficiency of IoT nodes' communication with SQL and NoSQL databases. Despite remarkable potential, this study suffers from several limitations. These limitations have been discussed in this section.

A. LIMITED NUMBER OF DATABASES

This analysis studies only two databases. There are many other popular SQL and NoSQL databases that have not been explored in this paper. It is a significant limitation of this paper. The API integration complexity is the reason behind this limitation. The researchers of this paper are conducting more studies on other SQL and NoSQL databases which will be published in the future.

B. SINGLE-MASTER MODE

The experimented cloud server offers both single-master and multi-master configurations. For the simplicity of the experiment, the single-master mode has been used in this research. It is one of the weaknesses of this paper. Selecting the benchmarks to evaluate the performance of NoSQL and SQL databases in multi-master configurations is challenging. This challenge will be explored in the future scope of this research [38].

C. EXPERIMENTAL ENVIRONMENT

The analysis presented in this paper is the findings on the experimental environment. It is beyond the scope of the researcher of this paper to conduct the research in an active industrial environment. As a result, the challenges of real-time industrial production have not been included in the analysis.

The limitations of this paper are considered as the future scope to conduct further research and discover more valuable data that will further contribute to improving the IoT environment efficiency.

VII. CONCLUSION

There are multiple solutions available for almost every technological aspect in this age of technological revolution. It facilitates a competitive market where the service providers compete with each other which ensures sustained and continuous development of the services and products. However, multiple options lead to another challenge which is choosing the most effective solution for a problem. Choosing an effective database management system to store voluminous IIoT data is one such problem. This paper has explored both SQL and NoSQL databases to discover the most effective and efficient option. This paper explores AWS Aurora SQL Database and AWS DynamoDB NoSQL Database which are SQL and NoSQL databases, respectively. The most common and popular protocol for IoT-database communication, MQTT has been used in this paper. The methodology has been designed carefully to ensure the effectiveness of the study. After carefully processing the data, a flawless test plan has been devised in this paper. A proper experimental setup has been designed to execute the test plan. The experimental data collected during the experiment shows that the NoSQL database reduces the latency. To cross-validate the findings, the same experiment has been conducted with 1, 10, and 100 units. In every case, the result shows that the latency of the SQL database is higher than the NoSQL database for IIoT. Further, the novel energy efficiency model designed and analyzed in this paper demonstrates that NoSQL is more energy efficient than SQL database for IIoT communication. Despite the valuable contribution of this study to facilitate a data-driven decision-making approach while choosing an effective database management system to store IIoT data, there are some limitations of this study. The experimental setup has been developed using a limited number of databases in a single-master model service. Whether the proposed methodology and its findings are valid for a large number of databases has not been addressed in this study. This limitation will be overcome in the subsequent research effort. Despite the limitations, this analysis unearths valuable information related to IIoT and database communication which has the potential to increase production efficiency and reduce operational costs by guiding the decision-making process of selecting an appropriate storage system to store sensor data of autonomous manufacturing facilites.

REFERENCES

[1] G. Sharma, V. Tripathi, and V. Singh, "A systematic analysis of trending NoSQL database tools and techniques: A survey," in *AIP Conf. Proc.*, vol. 2782, no. 1, 2023, Art. no. 020091.

[2] M. Nasereddin, A. ALKhamaiseh, M. Qasaimeh, and R. Al-Qassas, "A systematic review of detection and prevention techniques of SQL injection attacks," *Inf. Secur. J., Global Perspective*, vol. 32, no. 4, pp. 252–265, Jul. 2023.

[3] T. Shareef, K. Sharif, and B. Rashid, "A survey of comparison different cloud database performance: SQL and NoSQL," *Passer J. Basic Appl. Sci.*, vol. 4, no. 1, pp. 45–57, Feb. 2022.

[4] P. S. Mehta, "NoSQL databases in Kubernetes," Tech. Rep., 2023.

[5] N. Bansal, S. Sachdeva, and L. K. Awasthi, "Are NoSQL databases affected by schema?" *IETE J. Res.*, pp. 1–22, Jul. 2023.

[6] L. F. da Silva and J. V. F. Lima, "An evaluation of relational and NoSQL distributed databases on a low-power cluster," *J. Supercomput.*, vol. 79, no. 12, pp. 13402–13420, Aug. 2023.

[7] R. Andreoli, T. Cucinotta, and D. B. D. Oliveira, "Priority-driven differentiated performance for NoSQL database-as-a-service," *IEEE Trans. Cloud Comput.*, early access, Jul. 4, 2023, doi: 10.1109/TCC.2023.3292031.

[8] C. Gomes, M. N. de O. Junior, B. Nogueira, P. Maciel, and E. Tavares, "NoSQL-based storage systems: Influence of consistency on performance, availability and energy consumption," *J. Supercomput.*, vol. 79, no. 18, pp. 21424–21448, Dec. 2023.

[9] M. Arshad, M. N. Brohi, T. R. Soomro, T. M. Ghazal, H. M. Alzoubi, and M. Alshurideh, "NoSQL: Future of bigdata analytics characteristics and comparison with RDBMS," in *The Effect of Information Technology on Business and Marketing Intelligence Systems*. Cham, Switzerland: Springer, 2023, pp. 1927–1951.

[10] S. Tantiphuwanart, N. Tuaycharoen, D. Wanvarie, N. Pratanwanich, and A. Suchato, "Performance improvement on a learning assessment web application using AWS DynamoDB as a cache database," in *Proc. 20th Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE)*, Jun. 2023, pp. 303–308.

[11] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, "SQL and NoSQL database software architecture performance analysis and assessments—A systematic literature review," *Big Data Cognit. Comput.*, vol. 7, no. 2, p. 97, May 2023.

[12] S. Khan, X. Liu, S. A. Ali, and M. Alam, "Bivariate, cluster, and suitability analysis of NoSQL solutions for big graph applications," in *Advances in Computers*, vol. 128. Amsterdam, The Netherlands: Elsevier, 2023, pp. 39–105.

[13] S. Kim, Y. Hoang, T. T. Yu, and Y. S. Kanwar, "GeoYCSB: A benchmark framework for the performance and scalability evaluation of geospatial NoSQL databases," *Big Data Res.*, vol. 31, Feb. 2023, Art. no. 100368.

[14] S. Mukherjee, "Benefits of AWS in modern cloud," 2019, *arXiv:1903.03219*.

[15] M. Elhemali et al., "Amazon $*DynamoDB*$: A scalable, predictably performant, and fully managed $*NoSQL*$ database service," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2022, pp. 1037–1048.

[16] M. A. Hoque, M. Hossain, S. Noor, S. M. R. Islam, and R. Hasan, "IoTaaS: Drone-based Internet of Things as a service framework for smart cities," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12425–12439, Jul. 2022.

[17] R. Meenaakshi Sundhari, K. Jaikumar, and L. Murali, "Message queuing telemetry transport (MQTT) protocol-based monitoring and segregation of wastes in smart cities using IoT," in *Advances in Data and Information Sciences*. Cham, Switzerland: Springer, 2022, pp. 487–494.

[18] N. Faruqui, M. A. Yousuf, M. Whaiduzzaman, A. K. M. Azad, A. Barros, and M. A. Moni, "LungNet: A hybrid deep-CNN model for lung cancer diagnosis using CT and wearable sensor-based medical IoT data," *Comput. Biol. Med.*, vol. 139, Dec. 2021, Art. no. 104961.

[19] N. Faruqui, M. A. Kabir, M. A. Yousuf, Md. Whaiduzzaman, A. Barros, and I. Mahmud, "Trackez: An IoT-based 3D-object tracking from 2D pixel matrix using Mez and FSL algorithm," *IEEE Access*, vol. 11, pp. 61453–61467, 2023.

[20] S. Achar, N. Faruqui, M. Whaiduzzaman, A. Awajan, and M. Alazab, "Cyber-physical system security based on human activity recognition through IoT cloud computing," *Electronics*, vol. 12, no. 8, p. 1892, Apr. 2023.

[21] L. P. O. Paula, N. Faruqui, I. Mahmud, Md. Whaiduzzaman, E. C. Hawkinson, and S. Trivedi, "A novel front door security (FDS) algorithm using GoogleNet-BiLSTM hybridization," *IEEE Access*, vol. 11, pp. 19122–19134, 2023.

[22] S. Trivedi, T. Anh Tran, N. Faruqui, and Md. M. Hassan, "An exploratory analysis of effect of adversarial machine learning attack on IoT-enabled industrial control systems," in *Proc. Int. Conf. Smart Comput. Appl. (ICSCA)*, Feb. 2023, pp. 1–8.

[23] S. Bhardwaj, S. Harit, and D. Anand, "Message queuing telemetry transport-secure connection: A power-efficient secure communication," *Int. J. Sensor Netw.*, vol. 42, no. 1, pp. 29–40, 2023.

[24] S. Trivedi, N. Patel, and N. Faruqui, "NDNN based U-Net: An innovative 3D brain tumor segmentation method," in *Proc. IEEE 13th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2022, pp. 538–546.

[25] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *J. Experim. Social Psychol.*, vol. 49, no. 4, pp. 764–766, Jul. 2013.

[26] S. Matam and J. Jain, *Pro Apache JMeter: Web Application Performance Testing*. New York, NY, USA: Apress, 2017.

[27] H. H. Azeez and M. Z. Abdullah, "Performance analysis of constrained application protocol (CoAP)," *AIP Conf. Proc.*, vol. 2591, no. 1, 2023, Art. no. 030074.

[28] R. Chard, K. Chard, K. Bubendorfer, L. Lacinski, R. Madduri, and I. Foster, "Cost-aware elastic cloud provisioning for scientific workloads," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 971–974.

[29] M. Sakr and A. Sadhu, "Visualization of structural health monitoring information using Internet-of-Things integrated with building information modeling," *J. Infrastruct. Intell. Resilience*, vol. 2, no. 3, Sep. 2023, Art. no. 100053.

[30] L. Malina, G. Srivastava, P. Dzurenda, J. Hajny, and R. Fujdiak, "A secure publish/subscribe protocol for Internet of Things," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, Aug. 2019, pp. 1–10.

[31] M. Shah, A. Kothari, and S. Patel, "A comprehensive survey on energy consumption analysis for NoSQL," *Scalable Comput., Pract. Exper.*, vol. 23, no. 1, pp. 35–50, Apr. 2022.

[32] K. Dineva and T. Atanasova, "Design of scalable IoT architecture based on AWS for smart livestock," *Animals*, vol. 11, no. 9, p. 2697, Sep. 2021.

[33] L. Shu, V. Piuri, C. Zhu, X. Chen, and M. Mukherjee, "IEEE access special section editorial: Convergence of sensor networks, cloud computing, and big data in industrial Internet of Things," *IEEE Access*, vol. 8, pp. 210035–210040, 2020.

[34] Z. Qian, J. Wei, Y. Xiang, and C. Xiao, "A performance evaluation of DRAM access for in-memory databases," *IEEE Access*, vol. 9, pp. 146454–146470, 2021.

[35] M. Tahmassebpour, "A new method for time-series big data effective storage," *IEEE Access*, vol. 5, pp. 10694–10699, 2017.

[36] B. Mishra and A. Kertesz, "The use of MQTT in M2M and IoT systems: A survey," *IEEE Access*, vol. 8, pp. 201071–201086, 2020.

[37] H. Siddharthan, T. Deepa, and P. Chandhar, "SENMQTT-SET: An intelligent intrusion detection in IoT-MQTT networks using ensemble multi cascade features," *IEEE Access*, vol. 10, pp. 33095–33110, 2022.

[38] D. Gamero, A. Dugenske, T. Kurfess, C. Saldana, and K. Fu, "SQL and NoSQL databases for cyber physical production systems in Internet of Things for manufacturing (IoTfM)," in *Proc. Manuf. Processes; Manuf. Syst., Nano/Micro/Meso Manuf., Quality Rel.*, vol. 2, Jun. 2021, Art. no. V002T07A014.

SOUKAINA BAKHAT KENITAR received the bachelor's degree in computer engineering from the National School of Applied Sciences, Tangier, Morocco, and the Diploma degree in data engineering from the Data Science School, Paris, France. She is currently pursuing the Ph.D. degree in quality of service for Internet of Things (IoT) with the National School of Applied Sciences (ENSA), Tetouan, Morocco.

MOUNIR ARIOUA is currently an Associate Professor with the National School of Applied Sciences, Abdelmalek Essaadi University. His research interests include wireless sensor networks, wireless communications, mobile computing, real-time processing and embedded systems, and the Internet of Things. His disciplines: computer engineering, computer communications (networks), and telecommunications engineering.

MOHAMMED YAHYAOUI is currently pursuing the Ph.D. degree with the Faculty of Science, Abdelmalek Essaadi University.

He is also a member of the Information Technologies and Systems Modeling Research Laboratory (TIMS), Abdelmalek Essaadi University. His research interests include computer science, including information technology, computer systems, data warehousing, OLAP, indexing, query optimization, star join queries, and bitmap join indexes. In addition to his academic pursuits, he is actively involved in organizing the scientific research days, a significant event hosted by Abdelmalek Essaadi University in collaboration with the Ministry of Higher Education, Scientific Research, and Innovation. He has contributed to the success of this event in multiple editions, including the 18th, 19th, 20th, and 22nd editions.