International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 13 (2021) pp. 124-135

Received: 27 Jan 2021; Accepted: 12 May 2021; Published: 25 July 2021

Database Performance Evaluation and Applications of Data Science for IoT Platform Analysis

Ha Duyen Trung

School of Electronics and Telecommunications,
Hanoi University of Science and Technology
trung.haduyen@hust.edu.vn

***Abstract*: Big data is very important for businesses and society because for the competitiveness of organizations, properly storing and managing large amounts of data enhances insight, decision-making, and process automation. Application of large storage technology, data mining and analysis is a research direction with practical needs when testing on IoT data sets. Deciding which tool to choose depends on the nature of work. One of the ways to evaluate the system towards data is to conduct the test in environment which the database will run, under the predicted data and the user's current workload. This paper comparatively studies on performance of opensource databases presenting in Internet of Things (IoT) systems such as NoSQL database-based MongoDB and SQL database-based PostgreSQL. Data stored in the database will be exploited, processed, and analyzed. Mining and analyzing data are one of the key parts in the data science. Data science is an interdisciplinary field of study that encompasses processes and systems for extracting knowledge within data in various structured and unstructured forms. Data science includes many areas of data analysis such as statistics, machine learning, data mining, predictive analysis. To provide the necessary rationale for the plans, it is to build a predictive model. Without predictive analytical tools we would be overwhelmed with data without information, not knowing what happened next. Based on the content, method and purpose of the forecast is divided into two categories: qualitative methods and quantitative methods. Qualitative methods often depend heavily on the experience of one or more experts in the relevant field. Quantitative method uses historical data over time, based on historical data to detect the movement direction of the object that is suitable for a certain mathematical model and at the same time use that model as estimation. This research work also presents analyzing and building time series prediction model.**

***Keywords*:** Internet of Things (IoT) analysis, Bigdata, Opensource, Database, Deep Learning**.**

I. Introduction

The Internet of Things (IoT) concept is both revolutionary as well as an enabler of automated and convenient daily life for modern day humans [1]. The development of the IoT can be accredited to a convergence in advances that took place over the past decade in computing, electronics, communications, and application design [2], [3]. IoT devices are designed with custom protocols that consider the resource constrained nature of these devices, to preserve power usage associated with device operations [4]. The most common IoT application-layer protocols are Constrained Application Protocol (CoAP), Message Queuing Telemetry Transfer (MQTT), Advanced Message Queuing Protocol (AMQP), Data Distribution Service (DDS), and Hyper Text Transfer Protocol (HTTP) [5].

Existing computation and storage paradigms comprise of cloud, fog, and edge computing. Through integration of these paradigms with the IoT, a robust data collection, storage, processing, and analytics framework emerges. Such a framework can provide real-time insights into data patterns and facilitates the application of machine learning techniques for realizing intelligent data analytics for the IoT. The cloud paradigm is a centralized model of data storage, that provisions various services such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), to enable processing and analysis of IoT data, centrally [6].

SQL-based relational databases provide centralized control of data, redundancy control and elimination of inconsistencies. The complexity and variability of Big Data require alternative models of databases. Primarily motivated by the issue of system scalability, a new generation of databases known as NoSQL is gaining strength and space in information systems [7]. NoSQL are database solutions that do not provide an SQL interface.

A four-tiered big data analytical engine is designed for the large amounts of data generated by the IoT platform of smart cities [8], [9]. The Apache Hadoop platform performs extraordinarily when used in the context of analyzing larger datasets [9]. A Hadoop distributed file system in IoT-oriented data storage frameworks allows efficient storing and managing of Big Data [10]. A model that combines Hadoop architecture with IoT and Big Data concepts was found to be very effective in increasing productivity and performance in evolutionary manufacturing planning which is an example of Industry 4.0 [11].

Applications of data science in building management may be seen as a domain that connects the smart city and manufacturing domain, because their solutions may be applied

to both domains. A study in [12] predicted the occupancy level of a building by observing the temperature, CO2, air volume, and air conditioning data. Random Forest was employed to classify the data and yielded 95 percent accuracy in predicting the occupancy level of rooms within the building. Another study in [13] collected building sensor data such as temperature, humidity, gas, smoke, CO, and CO2 to equip the building with a early detection system against gas leakage and fire hazard. The study arranged four risk levels, from no risk to high risk; and the regression trees algorithm achieved 99.93 percent accuracy in predicting the risk levels.

In this paper, we study and analyze the role of database and machine learning to facilitate data analytics for the IoT platform. The rest of the paper is structured as follows. Section II presents IoT architecture and database systems including presenting IoT platform and its database system. We also present in this Section the data science of deep learning, RNN and LSTM for the IoT data analysis. Section III presents the system performance evaluation of database models and prediction results of monitored environmental data. Section IV concludes the paper.

II. IoT Architecture and Database Systems

An IoT architecture is basically represented by 5 parts: the perception layer, the communication layer, the middleware layer, the application layer, the business layer. The Figure 1 depicts the 5-layer IoT architecture.

- The perception layer. It known as the device layer, includes physical objects and sensing devices. The sensors can be RFID, 2D barcodes, or infrared sensors depending on the object recognition method. This class is basically concerned with identifying and obtaining specific information about the objects using the sensing device. Depending on the sensor type, the information can be about position, temperature, orientation, movement, speed, humidity, and chemical changes in the air. Information is then passed to the communication layer for secure transmission to the information processing system.
- Communication layer. It also be called transport layer or network layer. This layer securely transmits the information from the sensor to processing system. The transmission medium can be wired or wireless and the technology can be 3G, 4G, Wi-Fi, Bluetooth Low Energy, LoRa, Z-Wave, Zigbee depending on the sensor device. Thus, the network layer transmits information from the cognitive layer to the middleware layer.
- Middleware layer. Devices on the IoT deploy different types of services. Each device only connects and communicates with other devices that deploy the same type of service. This layer is responsible for service management and database linkage. It receives information from the network layer and stores it in a database.
- Application layer. It provides complete management of the application based on information of objects processed in the middleware layer. Applications deployed by IoT can be smart health, smart agriculture, smart home, smart city, smart transportation, etc. Application layer combines with protocols such as CoAP, MQTT, HTTP.
- Business layer. It manages all layers, activities, and services of the IoT system. It combines several charts, diagrams, and dashboards based on data obtained from the application layer. This class can determine efficiency for big data analysis.

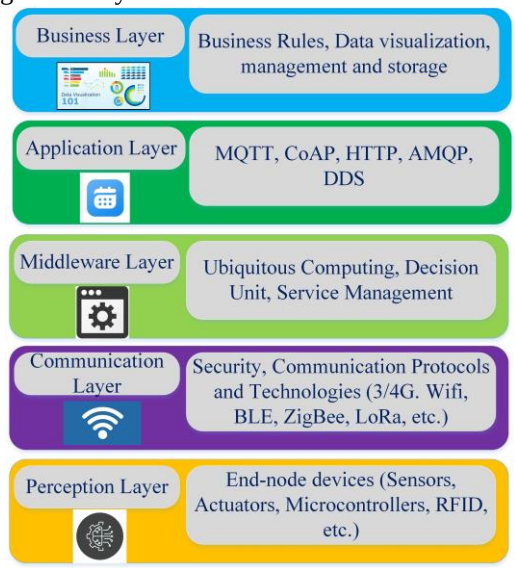

Figure 1. An architectural model of IoT 5 layers

A. IoT Platform

The IoT Platform is a core component of the overall IoT architecture. The IoT system provides product development tools to facilitate the application in a convenient, fast, and responsive way to a required IoT system. The IoT-based system plays the role of connecting devices, sensors, and IoT Gateway that allows them to exchange data securely in real-time, managing a huge number of devices.

IoT-based systems help manage strategies related to the connectivity of smart devices and their impact on business. IoT refers to the wireless communication between devices and their ability to send, receive, and create data based on user activity and environmental factors. IoT management products help businesses monitor and obtain communication between registered devices, as well as control devices from remote interfaces on computers or mobile devices. Logging and data storage products from connected smart devices provide real-time insights that help businesses spot trends and become more efficient.

For a system to qualify as an IoT-based system, it is necessary to synchronize and monitor the operation of smart devices, provides tools to control, update and retrieve data from synchronized devices, allows actions to be performed based on data received from the devices, and provide dashboard analysis for the device.

Metrics of an IoT platform-based systems are primarily determined by its flexibility, number of functions, stability, and security. The essential components of an IoT platform include the following:

- Device management: The devices do not always operate in a stable state. They need to be continually managed, reconfigured, updated as well as control settings. Edge devices act as a compiler and data gateway between networks. It creates a lot of operational problems and needs to be fixed manually. Therefore, remote access

control via the IoT platform is essential to the requirement of an IoT system. The device management architecture must be full-duplex communication (receiving information from IoT devices and sending control messages to IoT devices), flexible to allow all components in the network to be connected. close and able to communicate with each other. Because there are a lot of devices that need to be monitored as well as connected to the IoT based system. It is impractical to manually construct each device so that they can be interconnected. Therefore, streamlining the management of all devices is a very important factor in the IoT system.

- Full-duplex connectivity and flexibility: provide greater control over products, delivers constant updates, and provides faster response to problems that need to be addressed. A full-duplex connectivity platform can solve the problem through a software update.
- Data analysis: Data collected from IoT devices should be analyzed and screened to bring high value things in the management work as well as solve urgent problems for components in the IoT system. An IoT platform needs to provide a full range of means, thereby offering value and keeping the business from being overwhelmed with a lot of information and not knowing how to handle them. Therefore, data analysis, diagnostics, data visualization by graphs on display tables are essential requirements for an IoT-based system. Not all data received from devices needs to be processed in real-time. An IoT solution needs to be created that clearly separates which data streams need to be processed in real-time and which ones will be processed and valuable in the long run. This separation helps clarify the types of data for specific purposes, thereby helping to optimize the data processing capabilities of the IoT system.
- Availability, scalability, and reliability: Those problems always arise with any technology solution. Deployed technologies need to be carefully re-viewed to identify factors that can affect their adaptability, availability, scalability, and reliability. Distributed scaling technologies and solutions need the following aspects: data storage and computing power, continuous availability with backup solutions to ensure system downtime is minimal.
- Security and confidentiality: This are a major challenge for an IoT-based system. The system connects millions of devices as well as the IoT gateway, hence the safety and data security of the connections are top priority. An IoT platform must have security support across multiple points. IoT Gateway and connected devices need to be sophisticated and flexible enough to adapt security approaches. The device layer should support a high level of encryption for sending and receiving messages. The base system should provide authentication methods when connecting to gateways and other devices. Proven mechanisms such as TLS/SSL can be used to encrypt information during communication.

B. IoT Database System

The main task of IoT services is to collect, processing, and analyze data objects, to establish specifications and measurements. Therefore, database operability is crucial and makes sense for the storage and management of IoT data. The variety of database management systems today put users in a major dilemma about which system is best suited for each IoT service provided. Database Management System (DBMS) is software that interacts with end users, applications, and the database itself to collect and analyze data. In addition, the database administration system also provides users with functions to control read and write access, specify report creation, and analyze usage.

1) SQL relational database

Databases are an organized way of storing data efficiently to ensure fast and accurate data entry and extraction. A database has many organization ways, in which an effective way is organized according to the relational model and is called a relational database. Relational databases organize data in tables and relate to each other to minimize data redundancy while ensuring efficiency in data storage and retrieval [14].

The basic components of a relational database include Table of data and Relationship. Data tables are a major component of a relational database. The table contains data in which column / field that show properties of the data table, for example: name, address, etc. Row of data line consisting of related data or also known as a record, Cell of intersects the row and column is the place to store the data, and primary key of a field or combined field used to define records. A primary key has two properties that cannot be duplicated or null. A table can or may not have a primary key, but for ease of management, it is common to define primary keys for tables. Relationship is created between two tables to determine the relationship between the data fields of two tables. In a relationship database, there are three forms of relationships: 1-1 relation, 1-n relationship of the most common relationship in a database, and n-n relation in which a record in one table corresponds to many records in the other table and vice versa.

The basic command group of SQL includes DDL (Data Definition Language): CREATE to create new tables, table views and other objects, ALTER to edits existing data objects, such as tables, and DROP to delete the entire table, view of the table or other objects in the database. DML (Data Manipulation Language): SELECT to extract specific records from one or more tables, INSERT to insert new data into the database, UPDATE to modify, update data in database, DELETE to delete data from the database, DCL for Data Control Language, GRANT to grant privileges to the user, and REVOKE to retrieve the rights granted to the user.

2) PostgreSQL database

PostgreSQL is SQL database. It is supported by a community of experienced developers who have made a huge contribution to making it a highly reliable DBMS system. PostgreSQL supports advanced data types and enhanced performance optimization, features only available in commercial databases such as Oracle and SQL Server. Compared with other relational database management systems, PostgreSQL supports strong object-oriented and relational database functions. Examples include complete support for reliable transactions such as Atomicity, Consistency, Isolation, and Durability (ACID). Below are the main features of

PostgreSQL:

- SQL: a structured query language for defining, accessing, and manipulating databases. Postgres uses a native SQL called PL / pgSQL (procedural language / postgreSQL). The big difference is that Postgres can perform more complex queries than SQL.
- Transactions: Postgres transactions comply with the ACID principle. Its data validity is quite reliable.
- Table: due to relational structure database, the exact element that determines the difference between PostgreSQL and the MongoDB mentioned below is that the entire schema needs to be designed and configured upon creation. Changing the table after it starts can be done, however may lead to errors that are not easily detected.
- Foreign key: PostgreSQL database can use foreign key. Foreign keys allow to keep their data normalized by referencing an object from one table to another, hence the second table has access to the keys and values of the first table.

In addition, PostgreSQL also has other features: it is compatible with various platforms using all major languages and middleware. It provides a sophisticated locking mechanism. It supports simultaneous control of multiple versions and complies with ANSI SQL standards. It is full support for client-server network architectures. It has SSL server redundancy and high availability. It supports JSON. Some advantages of PostgreSQL can be mentioned as: PostgreSQL can run dynamic websites and web apps as LAMP stack option, and it is a highly error-tolerant database.

PostgreSQL source code is available for free under opensource license. This allows the freedom to use, modify and deploy it according to business needs. PostgreSQL supports geographic features so that it can be used as a storage of geospatial data for location-based services and geographic information systems. PostgreSQL is used in many applications such as: financial industry, manufacturing, web technology and NoSQL, scientific data, etc. PostgreSQL also uses basic SQL statements like UPDATE, SELECT, INSERT, ORDER BY, DELETE queries in conjunction with clauses like WHERE, WITH, LIKE, AND & OR.

3) NoSQL relational database

NoSQL is an open-source database that does not use relational model (RDBMS). The term NoSQL marks the development of a new generation of databases: distributed and non-relational. It is possible to expand data without worrying about things like creating foreign keys, primary keys, and checking constraints. Besides, NoSQL ignores data and transaction integrity in exchange for fast performance and scalability.

In NoSQL, data can be stored schematically or freely. Any data can be stored in any record. Among NoSQL databases, there are 4 popular data storage models: document database, key-value stores, wide column storage and graph database. In document database (e.g., CouchDB, MongoDB), documentation is the main principle of database data types. The document describes itself, inheriting from a tree data structure. It can be said that the document database is part of the key-value. The added data is stored as a free JSON structure or document, where the data can be of any type, from integers to strings or to free documents. Key-value stores (e.g., Redis, Riak) is the simplest NoSQL type of data storage using from an API. We can get a value for a key, set a value for a key, or delete a key from the data. Stored value pairs are always accessible via primary key and generally have good access performance. Wide column stores (e.g., HBase, Cassandra) is stored in columns instead of rows like in a conventional SQL system. Any number of columns (and therefore a variety of data types) can be grouped or aggregated as needed for data queries or views. Graph database (e.g., Neo4j) is represented as a network or graph of entities and the relationships of that entity, with each node in the graph being a free-form block of data.

The structured query language used by traditional databases provides a unified way of communicating with the server when storing and retrieving data. Highly standardized SQL syntax. In contrast, each NoSQL database has its own syntax for querying and managing data. For example, CouchDB uses requests in the form of JSON, sent over HTTP to create or retrieve documents from its database. MongoDB sends JSON objects over a binary protocol using a command line interface or language library.

4) MongoDB database

MongoDB is a document-oriented NoSQL database. Therefore, MongoDB will avoid the relational database's table-based structure to accommodate documents such as JSON that has a very flexible schema called BSON. MongoDB uses to store data in the form of JSON documents, so each collection will have different sizes and different documents. The data is stored in JSON document; hence the query is very fast.

With relational database we have the concept of tables, relational databases (e.g., MySQL or SQL Server, etc.) use tables to store data, with MongoDB we will use the concept of collection instead of tables. Collections in MongoDB are structured very flexibly, allowing data to be stored without following a certain structure. Common terms used in MongoDB can be explained as:

- _id: is a required field in each document. The "_id" field represents a unique value in the MongoDB document. The _id field can also be interpreted as the primary key in the document. If you add a new document, MongoDB will automatically generate a "_id" representing that document and is unique in the MongoDB database.
- Collection: group of documents in MongoDB. Collection can be understood as a corresponding table in the RDBMS (Relational Database Management System). Collection resides in a single database. Collections do not have to define columns, rows, or data types first.
- Cursor: is a pointer to the result set of a query. The client can iterate over a pointer to get the results.
- Database: the place containing the Collection, like the RDMS database they contain the tables. Each Database has its own file stored in physical memory. A few MongoDB owners can hold multiple databases.
- Document: a record belonging to a Collection is called a Document. Documents contain name and value fields, respectively.
- Field: is a key - value pair in a document. A document can

have zero or more fields. The fields are like columns in a relational database.

- JSON (JavaScript Object Notation): readable in a plain text format representing structured data. JSON currently supports many programming languages.
- Index: are special data structures, used to hold a small portion of data sets to easily scan. An index stores the value of a particular field or set of fields, sorted by their values. Index effectively supports the analysis of queries. Without an index, MongoDB would have to scan all documents of the collection to select which documents match the query. This scan is inefficient and requires MongoDB to process large volumes of data.

Architecture of MongoDB

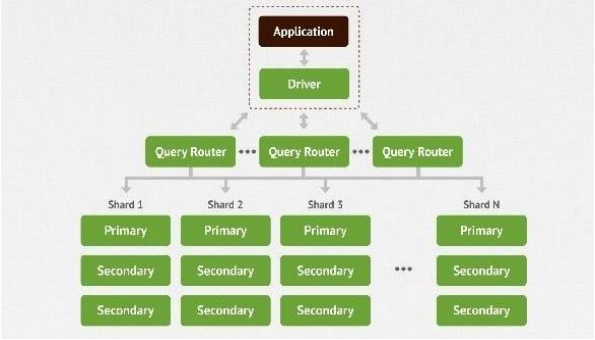

Figure 2. MongoDB Architecture.

MongoDB works under an implicit process, always opening a port (default port is 27017) to listen for query requests, manipulate from sending applications and then proceed.

Each MongoDB record is automatically appended with a field named "_id" of the Object Id data type that it specifies to determine the uniqueness of one record relative to another, as well as for other operations. Data field "_id" is always automatically indexed for the highest speed of querying information.

Every time there is a data query, the record is cached (buffered) to the RAM memory, so that the following query can be served faster without having to read it from the hard drive. When there is a request to add / edit / delete records, to ensure the performance of the default application MongoDB will not update to the hard drive immediately. However, after 60 seconds, MongoDB will write all data changes from RAM down to hard drive. MongoDB uses basic commands like UPDATE, FIND, INSERT, REMOVE documents.

MongoDB main features

- Aggregation framework: Aggregation operations in MongoDB process the data record data and return the calculation result. It is built on the idea of information processing pipes where the documents are treated as input of aggregate data. MongoDB also provides electronic schematic reduction operations support for fast data aggregation.
- Horizontal scalability: MongoDB supports horizontal scaling, disseminating data across multiple machines, thus helping facilitate high throughput on large files. Shading allows adding additional versions to expand the capacity of the database when required.
- Supports multiple archiving tools: The storage engine in MongoDB ensures the way data is put into memory and on disk. MongoDB offers the freedom to choose different storage engines according to the application's requirements as any single tool may not be the best solution for all types of uses.
- Document indexing: Indexes stores collection data sets in an easily browsable structure. MongoDB supports single, concatenation, multikey, geospatial, and hashed indexes on data, helping to discover documents that match query requirements without performing a full crawl scan.
- Replication: This is a very important feature of MongoDB and is recommended to use when installing MongoDB for real environment. Replication is copying a data object to a group of different servers. One server in this group of servers will be the primary mirror server and the other servers as secondary replication. The main mirror server acts as the administrator, where all logging and data object updates need to go through that server. Secondary servers can be used for reading data and load balancing. MongoDB has automatic failover, when the primary replication server fails, one of the secondary servers will be elected as the primary server instead to ensure the write operations are always performed.

C. Data Science: Deep Learning, RNN and LSTM

Data science is an interdisciplinary field that encompasses processes and systems for extracting knowledge within data in different structured and unstructured forms. Data science includes many areas of data analysis such as statistics, machine learning, data mining, predictive analysis. Data science uses a variety of techniques and theories such as operations science, information science, computer science, signal processing, probability modeling, and statistics studies, database, pattern recognition, visual description, fuzzy logic, data warehouse, artificial intelligence, decision support system, intelligent business. Methods that move towards big data are particular interest to data science. Data science has many applications in life such as machine translation, speech recognition, search engine, biomedical, healthcare, social sciences and economics, finance, insurance.

The advancement of machine learning has made a significant contribution to the growth of data science. Machine learning is a field of computer science that provides computers with the ability to learn through data and experience without requiring explicit programming. As an area of data analysis, machine learning is a method that allows complex models and algorithms building to find the knowledge hidden within the data that support decision-making and business processes.

Recently, Artificial Intelligence (AI), specifically machine learning, has emerged as a testament to the industrial revolution 4.0, from economics, education, medicine to housework, entertainment or even military. Prominent applications in AI development come from many disciplines to solve a variety of problems. But most of the breakthroughs come from deep learning - a small array that is gradually expanding to each type of work, from simple to complex. Deep Learning has helped computers do things: categorize thousands of different objects in images, create annotations, communicate with people, even forecast weather. In the past few years, Deep learning has been mentioned a lot as new trend

of the AI revolution. Deep learning exploits Big Data with much higher accuracy than other machine learning methods on the data set, especially for images.

1) Recurrent Neural Network

Figure 3 shows a segment of the recurrent neural network of A with input x_t and output h_t. A loop allows information to be passed from one step to the next of the neural network.

A recurrent neural network can be thought of as multiple copies of the same network, where each output of one is the input to another copy network. This repeating chain of networks is the resolution of regression neural network, loops that make them form a list of duplicating networks. The network nodes still take input and have the same output as a pure neural network. A neural network is made up of single neurons called perception. A neural network is a combination of layers of perception, called as multilayer perception.

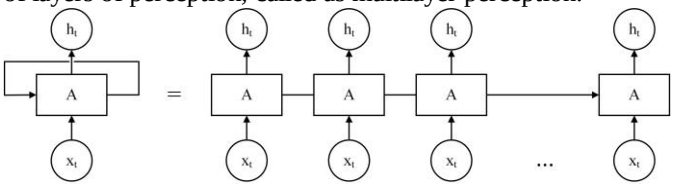

Figure 3. Resolution of the RNN

Over the years, researchers have developed many types of sophisticated regression neural networks to develop the traditional regression neural network model. Bidirectional RNN model has the output at step *t* depends not only on the front elements but also on the back elements. For example, to predict the missing word in a sentence, it is necessary to consider both the previous and the latter part of a sentence. Therefore, we can consider the model as superposition of two opposite RNNs on each other [15]. Now the output is calculated based on both latency states of these two opposite RNN networks. Deep Bidirectional RNN is like bidirectional RNN but differs in that they contain many hidden layers at each step. In practice, they help the higher levels of learning, but more training data is also required.

2) Long Short-Term Memory (LSTM)

LSTM is commonly a special form of RNN, capable of learning distant dependencies. Basically, the model of LSTM is not different from the traditional model of RNN, but they use different calculation functions in hidden states. The memory of the LSTM is called the cell and they take the front state h_{t-1} input and the current input x_t. It decides for itself what to remember or erase. Then, in conjunction with forward state, current remembers and current input. Hence, the problem of remote dependencies can be avoided. Memorizing information for long periods of time is their default feature, but we do not need to train it to be able to remember it, which means it itself can be memorized without any intervention.

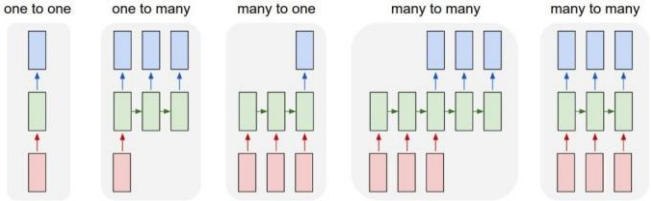

Figure 4. Classification of the LSTM model

Unlike standard feed-forward neural networks, the LSTM has feedback connections. It can not only process single data points but also entire data series. LSTM can apply next trend prediction of a data series. The classification of LSTM problem and choosing the right problem for data is also very important. LSTM works extremely effectively on many different problems, it therefore has gradually become popular [16]. LSTM can be classified into four characteristics as illustrated in Fig. 4 as follows:

- One-to-one: same problem pattern for neural networks with one input and one output.
- One-to-many: a problem has one input but many outputs, for example, the problem of annotating the image, the input is an image, however the output is many descriptive words for the image.
- Many-to-one: a problem with many inputs but only one output, for example, the problem of classifying actions in the video, the input is multiple images (frames) separated from the video, even though it is the action in the video.
- Many-to-many: a problem with many inputs and many outputs, for example, the problem of translating from English to Vietnamese, input and output is a multi-word sentence.

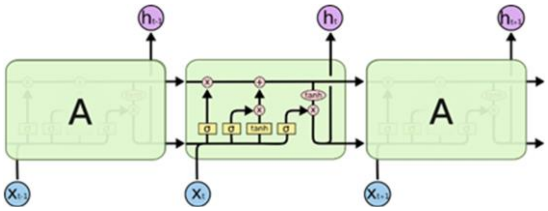

Figure 5. The iteration module in LSTM contains four layers of interaction

The Figure 5 shows the iteration module in LSTM containing four interaction layers. In this diagram, each line carries a vector from the output of one node to the input of another. The pinks represent mathematical operations such as vector addition, while the yellow plots are used for learning in individual neural networks. The chord lines denote the merging, while the fork lines indicate its content copied and moved to different places.

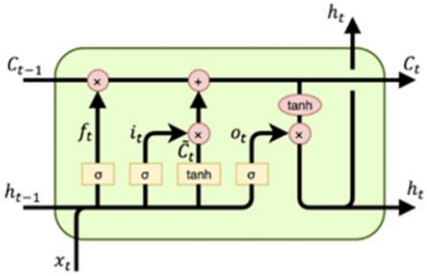

Figure 6. Module details of LSTM

An LSTM module has five essential components that allow it to model both long- and short-term data, including:

- Cell state (c_t): represents the cell's internal memory that stores both short-term and long-term memory.
- Stealth state (h_t): is the calculated output status information w.r.t. current input, previous hidden state, and current cell input that ultimately uses for future prediction. In addition, the stealth state may decide to retrieve only the short or long term memory types, or both types of memory stored in the cell state for further prediction.

- Input port (i_t): determines how much information from the current input passes to the cell state.
- Lost gate (f_t): determines the amount of information from the current input and the previous cell state enters the current cell state.
- Output port (o_t): determines the amount of information from the current cell state to the hidden state, therefore, if necessary, the LSTM can only select the long term or the short-term memory.

The first step of LSTM is to decide what information to remove from the cell state. This decision is made by the sigmoid layer-called "forget gate layer". It will take the input h_{t-1} and x_t and then return a number in the range [0, 1] for each number in the cell state c_{t-1}. Output "1" indicates that it holds all information, while "0" indicates that all information will be discarded.

$$f_t = \sigma\left(W_f[h_{(t-1)}, x_t] + b_f\right). \quad (1)$$

The next step is to decide what new information to store in the cell state. This has two parts. The first is to use a sigmoid layer called the "input gate layer" to decide which values we will update. Then, a tanh layer that creates a vector for the new value C_t adding to the state. Next, those two values will be combined to create an update to the state.

$$i_t = \sigma\left(W_i[h_{(t-1)}, x_t] + b_i\right). \quad (2)$$

$$\tilde{C}_t = \tanh\left(W_c[h_{(t-1)}, x_t] + b_i\right). \quad (3)$$

Then, it is to update old cell state C_{t-1} to new state C_t. Multiplying the old state by f_t to get rid of the information that decided to forget earlier, then add $i_t * \tilde{C}_t$. This newly obtained state depends on how we decide to update each state value.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \quad (4)$$

Finally, the output value will be decided based on the cell state. However, it will be further screened. First, running a sigmoid layer to decide which part of the cell state we want to output. Then, passing it in a subtle state through a tanh function to reduce its value to about [-1, 1] and multiply it by the output of the sigmoid gate to get the output.

$$o_t = W_0[h_{(t-1)}, x_t] + b_i. \quad (5)$$

$$t_t = o_t * \tanh\left(C_t\right). \quad (6)$$

3) Keras and Prediction Models

Keras: it is an opensource neural network written in the Python, containing a library developed by Francois Chollet for deep learning research engineers. Keras can be used together with well-known libraries such as Tensorflow, CNTK, Theano. Advantages of Keras such as easy to use, simpler to use than Tensor, fast modeling, works on both CPU and GPU, support building CNN, RNN or both, and simple, easier to grasp than other libraries in building models. Keras contains specialized layers with many classes to build models such as CNN, RNN, GANs. However, there is a few classes used in this work.

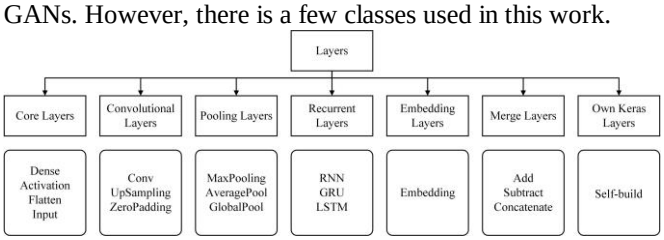

Figure 7. Structured layers to model Keras

Figure 7 presents the Keras structed layers. The core layer contains classes that almost any model uses it. Dense layer used as a normal neural network layer. Recurrent Layers contain concurrent layers. Activation is used to select the activation function in the class (can use an alternate activation parameter). Data pre-processing in Keras is divided into three parts: string, word, and image pre-processing. Functions in the skeleton of the model include loss, optimizer, activation models.

Predictive model: it can be a mathematical function that can map between a set of input data variables, usually encapsulated in a record, into a response or target variable to a certain predictor variable. In fact, a predictive model needs to have many independent variables selected from the data set. Besides that, processes, policies specific to each science sector must be combined. in the forecast model. The predictive model can be divided into two categories from the point of view of data miners, one is supervised learning, the other is called unsupervised learning.

When the learning process is supervised, during training the data is presented in a predictive model with the input and output data or desired outcome. Training process is repeated until the model learns mapping function between the given inputs and the desired output. To do this, a standard data set are usually created to be the training data set. Then, the data set will "train" through the specifying models. Parameters generated from the training dataset will be used to verify the model suitability on testing data set. There are two typical classes in regression and classification.

A predictive model can also make use of unsupervised learning. In this case, it is only presented with input data. Then its job is to find out how the different input records are related to each other. Clustering is the most used predictive model that uses unsupervised learning.

We can observe that the model is a combination of data and predictive modeling techniques. Thus, a predictive model is the result of combining data and mathematics, where learning can be translated into creating a mapping function between a set of input fields and the response or dependent variables.

Time series: a conventional machine learning data set is a set of observations and times that play a role in the common machine learning data set. Real-time series data is a series of values of a quantity defined over time. The time series values of the quantity X are $X_1, X_2, ..., X_n$, where X is the value of variable at time *t*.

The purpose of time series analysis is to understand or model the random mechanisms that generate an observed series and predict future values of series based on history of that chain. The time series modeling skill is determined by its performance in predicting the future. The time series are composed of four

components as:

- Long-term trend: This component is used to indicate an increase or decrease trend of quantity X in a long time. Graphically, it can be represented by a straight line or a smooth curve.
- Seasonal: This element indicates the seasonal increase or decrease of quantity X (can be based on month of year).
- Cyclical: This component indicates the change of quantity X periodically. It differs from the seasonal component in that the cycle of quantity X lasts more than one year. To evaluate this component time series values were observed annually.
- Irregular: This element refers to the irregular change of values in a time series. This change cannot be predicted by historical data, and it is not cyclical.
- Predictive error is the difference between real data values and the predicted values to assess the quality or suitability of the forecasting model at the same time.

Predictive errors are used to adjust the parameters of the forecasting model. Define ε_t is the forecast error at time t. Y and Y_t are respectively the actual value and the predicted value at time t. The forecast error can be expressed as

$$\varepsilon_t = Y_t - Y_t. \quad (7)$$

The predictive model is well evaluated when the forecast error is small. In addition, the randomness of error is also an important parameter to evaluate the accuracy of the forecast. When making forecasts one often assumes initial random data; calculations, evaluations and tests are also based on this assumption (random, normal distribution), so if the model is correct, the error must not be in any direction. There are four performance criteria to evaluate errors commonly used in prediction as follows:

Mean absolute error (MAE):

$$MAE = \frac{\sum_{t=1}^{n}|\varepsilon_t|}{n} = \frac{\sum_{t=1}^{n}|Y_t - Y_t|}{n}. \quad (8)$$

Mean squared error (MSE):

$$MSE = \frac{\sum_{t=1}^{n}\varepsilon_t^2}{n} = \frac{\sum_{t=1}^{n}\left(Y_t - Y_t\right)^2}{n}. \quad (9)$$

Mean absolute percentage error (MAPE):

$$MAPE = \frac{\sum_{t=1}^{n}\frac{|\varepsilon_t|}{Y_t}}{n} = \frac{\sum_{t=1}^{n}\frac{Y_t - Y_t}{Y_t}}{n}. \quad (10)$$

Root mean square error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}\varepsilon_t^2}{n}} = \sqrt{\frac{\sum_{t=1}^{n}\left(Y_t - Y_t\right)^2}{n}}. \quad (11)$$

Criteria of MAE and MSE and RMSE have the same characteristics and performance and often give the same evaluation results. However, the expert recommends that if the error values ε_t are equal, the MSE criterion should be selected for evaluation. In contrast, if the error values ε_t are too different, the MAE criterion should be selected for evaluation. The RMSE criterion is the square root of the MSE criterion, the difference is that the value of the RMSE criterion is less. The MAPE criterion relatively helps to evaluate errors. Assuming the average error of 1 unit compared with the data value of 100, it is still small (1%). Conversely, the average error of 1 value relative to the data value of 10 is considered large (10%). Therefore, when evaluating predictive errors with different data sets, the MAPE criterion should be used. In contrast, with the same data set but applying many different forecasting methods, MAPE should not be applied because of the complexity in calculation.

III. Performance Evaluation

A. *Methodology*

The experimental setup included a virtual machine created with VMware. The virtual machine has the hardware configuration: 4GB RAM, 1 CPU with two 2.8 GHz, and 160 GB hard drive. The operating system used to evaluate these databases is Ubuntu 16.04LST x64. This research work uses IoT data received from sensors mounted on IoT devices to measure the environmental air and water parameters for PostgreSQL up to 600,000 records. The database fields are from measurements of temperature, humidity, and turbidity of water, parameters of gases in the air environment such as CO, smoke. The original database was PostgreSQL and was ported to MongoDB using the data compiler. Database structure PostgreSQL was used from database of the "thingsboard" open-source IoT platform that contains a ts_{kv} table that stores environmental parameter data. In addition, to test with the YCSB validation tool, we create a user table containing 500K records. This user table stores 1KB records. The following tests were performed using the above IoT data. Each experiment was performed five times. Average response time query values have been estimated. Tools and software used to do the experiments were used including:

- Measure and retrieve data via IoT platform, store data in a PostgreSQL database called thingsboard.
- SQL database is PostgreSQL 12.4 and NoSQL is MongoDB 3.2.20.
- Using Python language and query syntax of each database.
- Mongify-Compiler data from SQL database to NoSQL.
- Benchmarking Cloud serving systems with YCSB benchmarking tool.

YCSB Benchmark: Yahoo Cloud Serving Benchmark is an opensource framework for evaluating and comparing the performance of big data systems. YCSB supports many different types of databases.

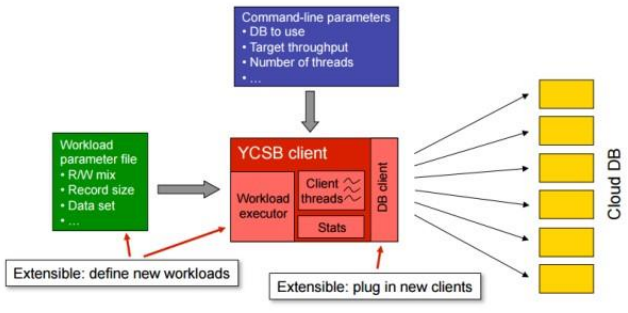

Figure 8. Activity diagram of YCSB calibration tool

Figure 8 illustrates the operation diagram of the YCSB calibration tool. The YCSB includes YSCB Client and Core workload. The Client is an extensible workload generator. Whereas the Core workload is a set of workload scripts. It provides an overall system's performance and YSCB Client allows declaring test scenarios specific to each system. These scenarios can be extended to measure the performance of many different types of databases. YSCB has libs that communicate with many popular database types. It is also possible to add other types of databases by proactively coding lib separately. To test the performance of different data types for comparison, multiple databases can be installed in the same deployment. Then it is possible to run the operations in turn on each different types of databases for evaluation. Steps to run a YCSB workload include:

- Set up test preparation database.
- Create table and load data, use YCSB manually with shell scripts that are specific to each database.
- Select workload in the YCSB.
- Set parameters according to the test scenario. Parameters to consider such as runtime, request allocation, number of concurrent running processes, etc.
- YCSB Client performs the workload.

The steps described above assume a single client server is running. For much larger clusters, it may be necessary to run multiple clients on different servers to generate enough load. Similarly, the database load may be faster in certain client usage scenarios. The YCSB includes a set of core workloads that define the underlying benchmark for cloud systems. Of course, it is possible to define their own workloads. However, the core workload is a useful first step, and taking these benchmark numbers for many different systems will allow to understand the balanced performance of different systems. In general, YCSB is a powerful performance testing tool, especially supporting many different types of big database. It is suitable for performance testing operations of simple workloads (can be deployed on many different databases) and propose the most suitable database selection for APP Bigdata performance.

B. Results and Discussions

In this section, we will deploy queries and processes against PostgreSQL and MongoDB databases. Results of every scenario implementation and each type of database are discussed. In addition, this section also provides results of running LSTM prediction model on IoT datasets.

1) Results of scenario experiments with databases

Scenario 1: Select query experiment.

The first scenario is to evaluate find-select queries, since IoT applications or data processors often use this type of query to interrogate databases and collect records for further evaluation. A fixed number of records are returned for the purposes of this test to measure query execution time between PostgreSQL and MongoDB. The total execution time of up to 500000 (500K) records returned in the IoT platform database is shown in the Figure 9.

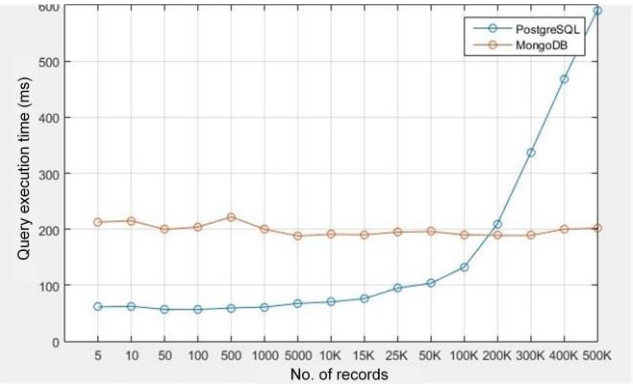

Figure 9. Select query for the PosgreSQL and MongoDB

Scenario 2: Insert query experiment

This script executes several insert queries in one process. The records size of 128 Bytes was used, formatted as JSON, which is the maximum data size employed in IoT applications. In the scenario, we did the experiment to estimate how many IoT devices can continuously transmit data to the database system of 500,000 records. The Figure 10 shows results of PostgreSQL and MongoDB in the insert query. MongoDB performs better for transactions with small number of insertion queries, compared to PostgreSQL. For medium or very large number of inserted queries (1,000-2,000 inserted queries), PostgreSQL outperforms MongoDB with a smaller average execution time.

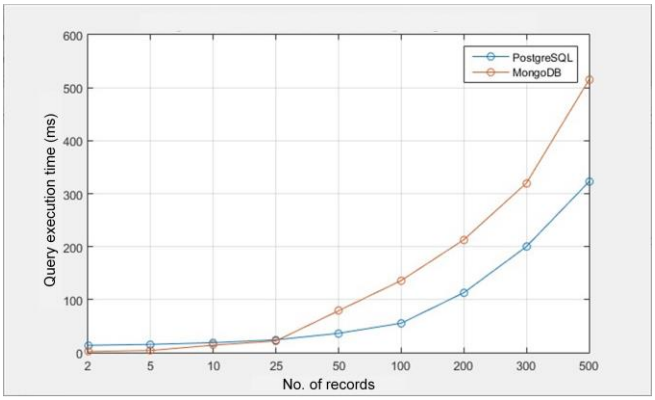

Figure 10. Insert query for the PosgreSQL and MongoDB

Scenario 3: Aggregation function experiment.

The first scenario was to evaluate find-select queries, since IoT applications or data processors often use this type of query to interrogate databases and collect records for further evaluation. In this scenario, the summary function is used as the averaging function.

For PostgreSQL database, the average aggregate function execution time with 500,000 records is about 0.1 seconds, this execution time is averaged with 10 execution times. For MongoDB database, the results of average parameters and execution time are shown in the Figure 11.

Figure 11. Test of Aggregation function

For an aggregate function applied on many IoT records, MongoDB is not a good choice for performing aggregate functions on IoT data. The execution time of MongoDB is much larger than that of PostgreSQL.

Scenario 4: Experiment of Read (Select) operation with YCSB.

The Figure 12 shows the query execution time and throughput for each number of PostgreSQL and MongoDB records. For read operations, MongoDB's throughput performing the number of processes per second is higher than that of PostgreSQL. Therefore, the execution time is faster. The document orientation system is faster than the relational system when performing many select operations.

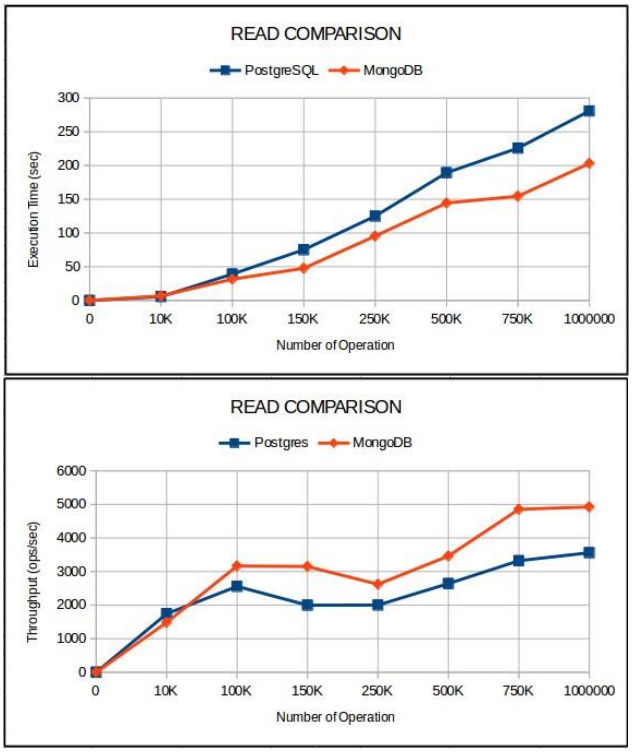

Figure 12. Experimenting the Read operation with YCSB

The Figure 13 shows the average latency and throughput achieved compared to the required throughput of PostgreSQL and MongoDB. It can be seen that PostgreSQL does not achieve the required throughput compared to MongoDB. The average reading latency of MongoDB is also lower than that of PostgreSQL. MongoDB shows better overall performance for read than PostgreSQL, as read and write is conducted in usable memory. It uses mapped files for data storage to achieve performance. PostgreSQL is better than MongoDB for lower record volumes. But rendering performance slightly lower than Mongo for performing read volume of larger records.

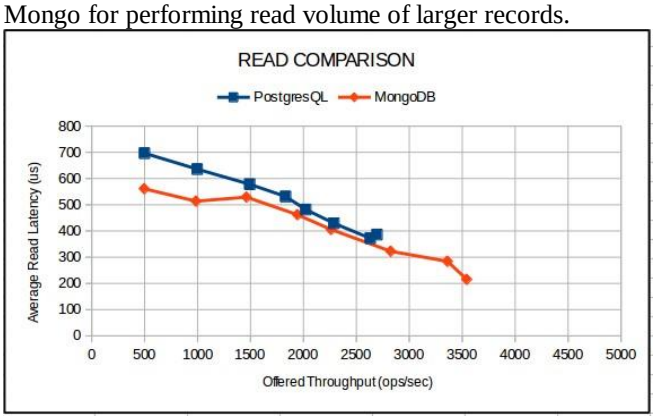

Figure 13. Test the Read operation with YCSB

Scenario 5: Experiment of the Insert operation with YCSB. The Figure 14 shows the execution time of Inserting by each number of PostgreSQL and MongoDB records. For insert queries and for small number of IoT records, MongoDB outperforms PostgreSQL, while for large number of records PostgreSQL has the least execution time compared to MongoDB.

Scenario 6: Experiment of the Update operation with YCSB. The Figure 15 provides the update execution time for each number of PostgreSQL and MongoDB records.

For the update operation, MongoDB's throughput performing the number of processes per second is about 60\% higher than that of PostgreSQL. Therefore, the execution time is also faster. For a large record volume update, MongoDB shows even more of its superiority in updating records.

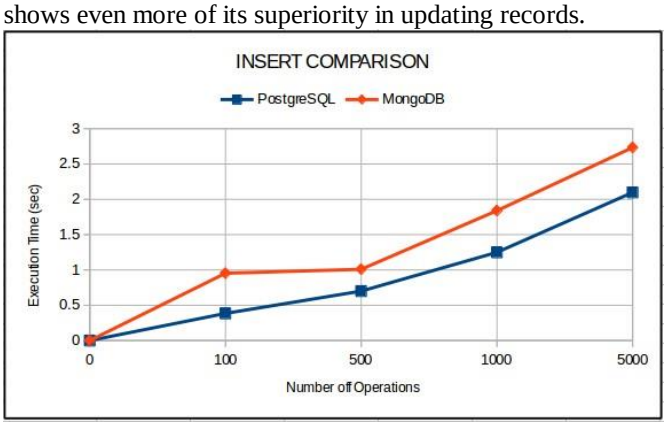

Figure 14. Experimenting Insert operation with YCSB

Moreover, it can be also seen from the Figure 16 that PostgreSQL does not achieve the required throughput compared to MongoDB. MongoDB's average update latency is also lower and more stable than PostgreSQL. MongoDB shows better performance for updating compared to PostgreSQL.

We have evaluated performance analysis of NoSQL and SQL databases by comparing PostregreSQL with MongoDB in terms of performance and scalability. MongoDB shows better performance metrics in most cases with large workloads. Hence, we can conclude that MongoDB is generally a better DBMS when consistency and atomicity are not the primary target. In transaction systems that consider consistency, SQL performs better than NoSQL.

Figure 15. Experimental UPDATE operation with YCSB

The advantages of MongoDB can be seen from tests performed using YCSB, which for each type of work shows that for all the parameters MongoDB performs better than MySQL. Especially in terms of lower latency and MongoDB throughput especially stands out for higher number of operations. Several related works have been reviewed and almost all show that NoSQL database is a better choice for large applications running in the cloud. In addition, YCSB is the good tool for running custom and standard workload databases, while evaluating performance, and that is very useful when making decisions about which database to choose business applications.

Execution time and Throughput vs. Number of operations. Relational databases suffer from inconvenient design, normalized forms, and types for IoT services, their limitations to maximum storage records, and their failure due to large data mainly requires the use of special categories. New and popular NoSQL databases are designed specifically for the IoT, as they provide collections without horizontal schemas, extremely useful for IoT data sourced from various sources of the structure, the hardware sense and the transmission protocol are different. According to the tests and results, for a small number of selected records, PostgreSQL outperforms MongoDB. MongoDB outperforms PostgreSQL on many selected records. For insert queries and for small number of IoT records, MongoDB outperforms PostgreSQL, while for large number of records PostgreSQL has the least execution time compared to MongoDB. Aggregate function execution tests have shown that PostgreSQL is the most suitable database system to perform aggregate functions on a small number of IoT data records. On the contrary, for an aggregate function applied on many IoT data records, MongoDB is not a good choice for performing aggregate functions on IoT data.

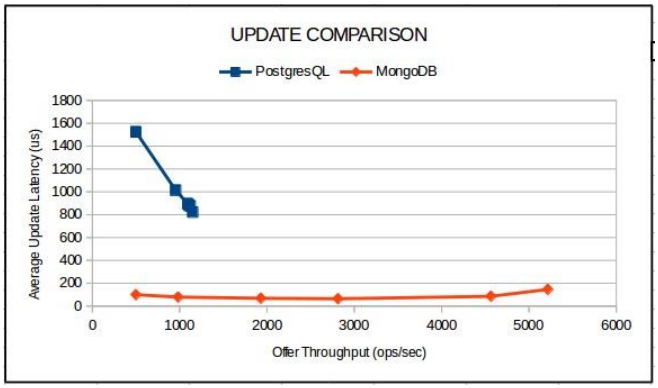

Figure 16. Experimental UPDATE operation with YCSB: Average update latency vs. Offer throughput.

2) Results of LSTM model on dataset

Running the LSTM model after detailed construction described above will plot the predicted and real values for comparison of every 24 hours in January of 2021. Here only sample the first five values and the last five values on the testing data set.

After taking steps to build a regression LSTM model for time series prediction, the results were able to generate predictions using the model for both testing and training datasets to get visualization of model's skills. The plotted data include the original data set in blue, the predictions of the training data set in green, and the predictions on the testing data set that are not visible in red. The following Figure 17 presents the results of running the LSTM model built on the temperature data set of Hanoi for 1 month and making the prediction from January 10 of 2021.

Figure 18 shown that the predicted values are closed to the expected real values. After running the LSTM model built above, we also estimate the RMSE results of all predictions.

The Table 1 presents accuracy of error estimation criteria between prediction and reality. The errors are measures of the good performance of a predictive model of the ability to predict the expected results of the regression problem. With such small error values and predictive results close to the expected values, the model can be considered as relatively accurate predictions.

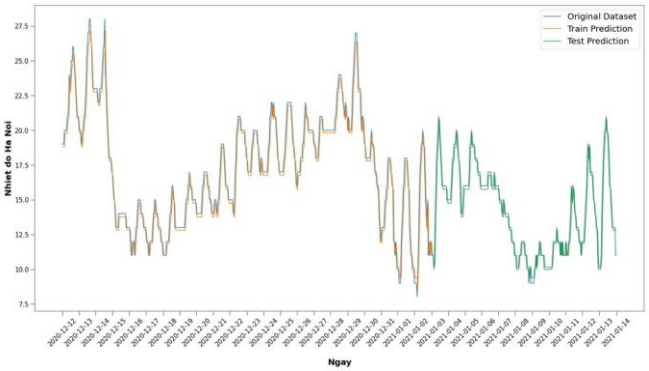

Figure 17. LSTM model on 1 month temperature data set in Hanoi

```
Date=2021-01-10 00:00:00, Predicted=9.813018, Expected=10.000000
Date=2021-01-10 01:00:00, Predicted=9.951144, Expected=10.000000
Date=2021-01-10 02:00:00, Predicted=10.019267, Expected=10.000000
Date=2021-01-10 03:00:00, Predicted=10.052782, Expected=10.000000
Date=2021-01-10 04:00:00, Predicted=10.064249, Expected=10.000000
Date=2021-01-17 02:00:00, Predicted=16.885990, Expected=17.000000
Date=2021-01-17 03:00:00, Predicted=16.886971, Expected=17.000000
Date=2021-01-17 04:00:00, Predicted=17.013681, Expected=17.000000
Date=2021-01-17 05:00:00, Predicted=17.067730, Expected=17.000000
Date=2021-01-17 06:00:00, Predicted=17.087747, Expected=16.000000
```

Figure 18. Samples of the first and last 5 values in the dataset.

Error estimation criteria	Accuracy
MAE	0.817
RMSE	0.962
MAPE	5.679

Table 1. Error values between prediction and reality.

IV. Conclusions

This paper comparatively studied performance evaluation of opensource databases in IoT systems such as NoSQL database-based MongoDB and SQL database-based PostgreSQL. Performance comparison of SQL and NoSQL databases has implemented and evaluated over the IoT platform. NoSQL database type was MongoDB is used to perform for NoSQL database whereas PostgreSQL was used to perform for SQL database. In addition, we also implemented the IoT data analysis, processing, especially the application of deep learning in building LSTM models for time series prediction for the data set stored in the IoT platform.

Acknowledgment

This research is funded by Hanoi University of Science and Technology (HUST) under project number T2020-SAHEP-015.

References

[1] A.K. Tyagi and A. Abraham. Internet of Things: Future Challenging Issues and Possible Research Directions. *International Journal of Computer Information Systems and Industrial Management Applications*, (12), pp. 113–124, 2020.

[2] H.D. Trung, N.X. Dung, N.H. Trung. Building IoT Analytics and Machine Learning with Open Source Software for Prediction of Environmental Data. In *Advances in Intelligent Systems and Computing*, vol. 1375, 2020, pp. 134–143.

[3] Ha.D. Trung, N.T. Hung, and N.H. Trung. Opensource Based IoT Platform and LoRa Communications with Edge Device Calibration for Real-Time Monitoring Systems. In *Advances in Intelligent Systems and Computing book series AISC*, vol. 1121, 2019, pp. 412–423.

[4] O. Aouedi, M.A.B. Tobji and A. Abraham. Internet of Things and Ambient Intelligence for Mobile Health Monitoring. *International Journal of Computer Information Systems and Industrial Management Applications*, (10), pp. 261–271, 2018.

[5] N. Naik. Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *Proceedings of 2017 IEEE international systems engineering symposium (ISSE)*, pp 1–7, 2017.

[6] H. Cao, M.Wachowicz, C. Renso, E. Carlini. Analytics everywhere: Generating in-sights from the internet of things. *IEEE Access*, (7), pp. 71749–71769, 2019.

[7] H. Vera, W. Boaventura, V. Guimaraes, and F. Hondo. (2015). Data Modeling for NoSQL Document-Oriented Databases. In *Proceedings of the CEUR Workshop*, pp. 129-135, 2015.

[8] A. Paul. IoT and Big Data towards a Smart City. In *Proceedings of the UGC Sponsored Two Day National Conference on Internet of Things*, Tamil Nadu, India, pp. 54–64, 2016.

[9] M.M. Rathore, P. Anand, A. Awais, and J. Gwanggil. IoT-based big data: from smart city towards next generation super city planning. *Int. J. Semant. Web Inf. Syst.*, 13(1), pp. 28–47.

[10] L. Jiang, L.D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu. (2014). An IoT-oriented data storage framework in cloud computing platform. *IEEE Trans. Ind. Inf.*, 10(2), pp. 1443–1451.

[11] S. Vijaykumar, S. Saravanakumar, and M. Balamurugan. Unique sense: smart computing prototype for industry 4.0 revolution with IOT and bigdata implementation model. *Indian J. Sci. Technol.*, 8(35), pp. 1–4, 2015.

[12] A. Dey, X. Ling, A. Syed, Y. Zheng, B. Landowski, D. Anderson, K. Stuart, M.E. Tolentino (2016) Namatad: Inferring occupancy from building sensors using machine learning. In *Proceedings of IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp 478–483, 2016.

[13] L. Salhi, T. Silverston, T. Yamazaki, and T. Miyoshi. (2019) Early detection system for gas leakage and fire in smart home using machine learning. In *Proceedings of IEEE International Conference on Consumer Electronics (ICCE)*, pp 1–6, 2019.

[14] N.M. Khushairi, N.A. Emran and A.K. Menon. Database Tuning Using Oracle Materialized View for Manufacturing Industry. *International Journal of Computer Information Systems and Industrial Management Applications*, (10), pp. 38–46, 2018.

[15] S. Bharati, Prajoy Podder, M. Rubaiyat Hossain Mondal. Artificial Neural Network Based Breast Cancer Screening: A Comprehensive Review. *International Journal of Computer Information Systems and Industrial Management Applications*, (12), pp. 125–137, 2020.

[16] M. Yu, F. Xu, W. Hu, J. Sun, and G. Cervone. Using Long Short-Term Memory (LSTM) and Internet of Things (IoT) for localized surface temperature forecasting in an urban environment. https://arxiv.org/abs/2102.02892.

Author Biographies

Ha Duyen Trung studied Engineer in Electronics and Telecommunications from Hanoi University of Science and Technology (HUST), Hanoi, Vietnam from 1998 to 2003, master and PhD in Communications Engineering from Chulalongkorn University, Bangkok, Thailand from 2003 to 2009, respectively. From 2007 to 2008, he was a research student at the Mobile Communications Laboratory (Araki Lab), Tokyo Institute of Technology, Japan. In 2012, Dr. Trung spent three months as a visiting scholar at Aizu University, Japan. He has been a lecturer at HUST since 2009. He joined HEEAP University Faculty Development Training at Arizona State University, AZ, USA in 2015. He published various research papers in the areas of IoT platform and related technologies, optical wireless communications including free-space optics (FSO) and visible light communications (VLC), signal processing for the next mobile communications, UAV communications, Satellite-based positionin, and navigation, and so on.