Cloud- Edge Network Data Processing based on User Requirements using Modify MapReduce Algorithm and Machine Learning Techniques

Methaq Kadhum[1], Saher Manaseer[2], Abdel Latif Abu Dalhoum[3]
King Abdullah II School for Information Technology
The University of Jordan, Amman, Jordan

***Abstract*—Edge computing extends cloud computing to enhancing network performance in terms of latency and network traffic of many applications such as: The Internet of Things (IoT), Cyber-Physical Systems (CPS), Machine to Machine (M2M) technologies, Industrial Internet, and Smart Cities. This extension aims at reducing data communication and transmission through the network. However, data processing is the main challenge facing edge computing. In this paper, we proposed a data processing framework based on both edge computing and cloud computing, that is performed by partitioning (classification and restructuring) of data schema on the edge computing level based on feature selection. These features are detected using MapReduce algorithm and a proposed machine learning subsystem built on user requirements. Our approach mainly relies on the assumption that the data sent by edge devices can be used in two forms, as control data (i.e. real-time analytics) and as knowledge extraction data (i.e. historical analytics).We evaluated the proposed framework based on the amount of transmitted, stored data and data retrieval time, the results show that both the amount of sending data was optimized and data retrieval time was highly decreased. Our evaluation was applied experimentally and theoretically on a hypothetical system in a kidney disease center.**

Keywords—Edge computing; cloud computing; data processing; data partitioning; MapReduce; machine learning; feature selection; user requirement

I. INTRODUCTION

Recently many applications seek to improve society by sharing distributed devices and processing their data, such as the Internet of Things (IoT) [1], Cyber-Physical Systems (CPS) [2], Machine to Machine (M2M) technologies [3], Industrial Internet [4], and Smart Cities [5].

However, there are great challenges to efficiently process, store, and manage the data collected by these applications. Since cloud computing (CC) has virtually an unlimited capacity in terms of storage and processing power; hence, cloud computing provides an opportunity to properly approach these tasks by integrating these networks and cloud computing [6].

To cope with these problems, Edge computing architecture has been proposed to support the CC applications and their requirements [7]. Edge computing can be considered as a layer or gateway between the applications' device's layer (edge devices) and the cloud layer (server in the cloud). This layer allows for speedy and direct processing of data produced by edge devices, rather than sending them to the central cloud infrastructure, enabling the CC applications to impose smaller latencies and to alleviate the traffic overhead on the network to the cloud [8]. Moreover, this layer has the power to process data and make intelligent decisions according to the data produced by the edge devices even before the data is processed by the cloud [7]. Consequently, the main functionality of an Edge Computing layer, when integrated with Cloud networks, is improving performance in terms of latency, and network traffic load through processing the data; hence, data processing is the main issue in edge computing [8]. Data processing in the edge layer is a relatively new topic, few studies have focused on this topic, although, there are still many open issues [9]. However, many researchers have addressed the issue of improving network performance in different ways to improve the quality of service and network performance in edge cloud networks [10]. Consequently, we observe that some of these researchers employed edge as a service [11] while others employed it to reduce the overload data with deep learning [12], and others use the edge as a fundamental tool [13]. Du B, et al. (2018) [11] designed Things-edge-cloud computing architecture to enable edge servers to cooperatively work with the cloud and achieve traffic-data as a service. Xu, X. et al. (2017)[14] presented edge Analytics as a service; rule-based analytics model equipped to edge node to prop the management of real-time analytic on edge. Alturki, B et al. (2017) [15] leveraged edge computing to provide low latency and network traffic by using confusion technique to preprocessing data in edge layer and Fu, J. et al, (2018) [13]proposed data processing schemes that combine fog computing and cloud computing to improve the quality of service in terms of latency, security, and flexibility and build object-oriented index to enhance data retrieval.

The main issues in CC that are faced by edge computing are storage, bandwidth, and real-time data analytics; this work presents a new method for improving edge cloud network's performance through data processing in the edge layer before sending it to the cloud. This is performed by processing the data schema on the edge computing level. Basically, the proposed technique of data processing is Data Partitioning, this procedure involves classifying the data and restructuring the data schema based on feature selection techniques. The objective of Data Partitioning is to partition the table into smaller tables in a manner, to allow the queries which access

only a fraction of the data to run faster because there are fewer data to scan, as well as to allow knowledge extraction data to be transmitted to the cloud, with low transmission time and low cost of bandwidth, because there are fewer data to load. Without loss of generality, we assume that smaller tablets are more likely to be located as close as possible to the CPU in the memory hierarchy, i.e, in the cache or the main memory.

Feature selection is a data preprocessing strategy based on dimension reduction, it directly selects a subset of relevant features, by removing irrelevant, redundant and noisy features from data; hence, preventing sending unnecessary data to the cloud to reduce the utilization of resources in the cloud (i.e. storage, bandwidth) as well as providing lower latency with data retrieval by only choosing relevant features [16], Feature selection techniques reduce storage and computational costs while avoiding significant loss of information [17]. Feature selection algorithms can be divided into wrapper, filter, and embedded methods. [18]. Wrapper Methods generate models with subsets of features and gauge their model performances. In this work, we presented a solution based on the forward search wrapper method.

In our proposed, features are detected using a modify MapReduce algorithm (M-MapReduce) and proposed a novel machine learning subsystem that are applied on user requirements (also called functional specifications in software engineering) [19]. These requirements reflect expectations for a new or modified product, requirements should be quantifiable, relevant and detailed.

The importance of this work is reflected from the main contribution of this paper, which is to develop a data processing framework based on both edge computing and cloud computing, by integrating the functions of data preprocessing that involves classifying, restructuring, storage, and retrieval. Therefore, this work focuses on the structure of the stored data as well as the transmitted data.

The infrastructure of the proposed framework consists of five main components as shown in Fig. 1: cloud-computing layer, edge computing layer, edge device layer, cloud computing queries, and edge device queries.

- Edge device collects data and then sends the data to the edge layer. In this work we assume that the data sent by edge devices can be used in both control data (i.e. real-time analytics) as well as knowledge extraction data (i.e. historical analytics). Control data is the streaming data that is processed in real time or time-limited data, on those data parts, real-time decisions are made e.g. monitoring, detecting fraud. While, Knowledge Extraction Data is the data stored for mining purposes and analyzing for support decision e.g. run reports, queries, and inferences from historical data [13].
- Edge layer where data is partitioned into control data (time-limited data) and knowledge extraction data; control data are extracted and processed in this layer. On the other hand, knowledge extraction data is uploaded to the cloud server layer. Thus edge layer is assumed to be stronger in both power and computing capability for model construction as well as the preprocessing and real-time analytics of the raw data sent by the edge device.
- Cloud computing layer for the data mining purpose and machine learning.
- Edge device queries are the search process in the device layer that is responsible for real-time decision making, constitute the queries in this layer (i.e. user requirements in devices layer such as control data). These queries are sent to a feature selection algorithm based on machine learning concepts, in order to extract key features that will assist in defining the real-time data.
- Cloud computing queries are the queries in the cloud layer that are executed for the mining process to extract knowledge from the data in the cloud layer (e.g. user requirements, data warehouse and data cube) that serves in other applications such as decision support systems. Moreover, these queries are also sent to the machine learning features extraction tool, in order to update feature extraction rules in order to accurately select features data to be sent to the cloud.

The proposed method in this paper fundamentally relies on the following assumptions:

First, the way we access data can help to restructure the data in a way that improves its locality characteristics such that the transaction time is reduced. In this work, we take note of the fact that data which exhibits good locality structure can be accessed in a much better way than data with the poor locality. By locality, we refer to a particular access pattern that can improve the efficiency of local memory (main memory or cache) in any system [20].

Second, the space-time product of a task or a set of tasks is inversely proportional to the throughput of the system [21].

Consequently, the main issues in the process of designing our framework are the detecting of accurate features for partitioning (classification and restructuring) the data without losing information, and implementing an efficient restructuring data algorithm. To illustrate our approach, we apply it on an imaginary system in a kidney disease center. We assumed that the edge cloud system is deployed in a chronic kidney disease center to monitor the patient's status and we use data from the UCI repository for chronic kidney disease (CKD) [22].

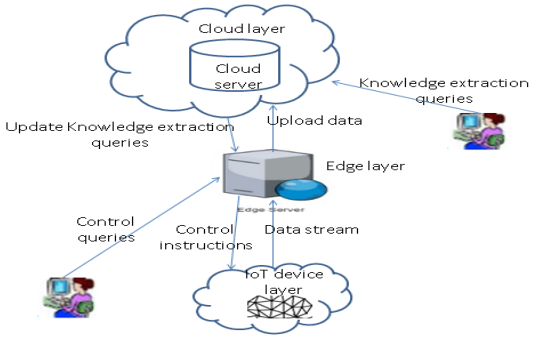

Fig. 1. The System of Preprocessing Data Classification, Restructuring, Storage and Retrieval.

The paper is organized as follows: Section 2 summarizes the edge computing, Section 3 presents the related work followed by a description of the proposed framework architecture in Section 4. Section 5 lists the details of the experiment environment and discusses the obtained results. Finally, Section 6 concludes the paper.

II. Edge Computing

With virtually unlimited computing power and storage resources, clouds are conceived to be the perfect platform for large-scale data analytics, while storage efficiencies also provide easy management of different applications. However, with cloud-based applications, most data must be sent to the data centers in the Cloud [23]. During the expansion of these applications, the volume of data increases and a large amount of data from these applications (e.g., IoT devices) are moved to the Cloud causing network bottlenecks due to bandwidth constraints. As time-sensitive and location-aware applications are developed (e.g., patient monitoring, real-time applications, transportation systems), the remote cloud will fail to fulfill the low-latency requirements of these applications; the round trip delay is too great [24].

Edge computing (e.g., Cloudlet, Mobile Edge Computing, and fog computing)[25][26] is proposed to overcome the problems faced by cloud-based applications [27] by offloading computing tasks to the edge of the cloud network. Using installed computing resources and intelligence at the network edge layer, a prompt response can be delivered to applications and the transmission of redundant data to remote clouds is avoided [28]. One more feature of edge computing is its distributed mode and support for device mobility inside heterogeneous network [29].

The Edge computing layer lies between cloud and end edge network devices; however, there is no commonly agreed-upon framework to capture the functionality of this computing paradigm. Recently, IBM and some researchers have proposed a three-layer paradigm as the high-level architectures, this is illustrated in Fig. 2 [30] [31].

Edge computing [32] is introduced for extending the Cloud Computing paradigm to the edge of the network to support many of cloud application (IoT, M2M, and CPS). Many characteristics have been found in edge computing, including low-latency, location awareness, mobility, and wide-spread geographical distribution, etc. These features make Edge computing suitable for different cloud applications.

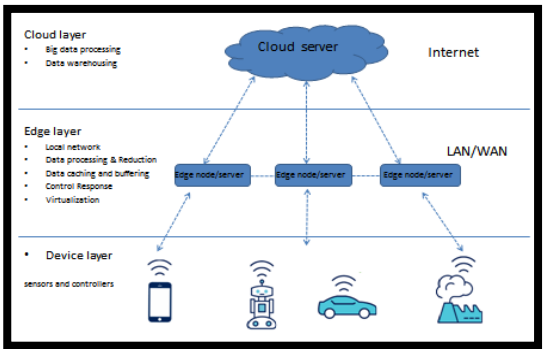

Fig. 2. An Example of an Edge Computing Architecture.

III. Related Work

Recently, many applications developed are internet-enabled such as surveillance, virtual reality, and real-time systems (e.g. monitoring)requiring fast processing and quick response time [33][34] . The core service and processing of these applications are performed on cloud servers; cloud computing provides an opportunity to these applications via an unlimited capacity in terms of storage and processing [6]. These applications produce a significant amount of data. This creates a number of challenges, such as high communication latency, and network traffic overhead, which is raised by transmitting all data to the cloud for processing and analysis. To overcome these challenges edge computing has been [35][36][37] proposed to extend the cloud computing paradigm to the edge of the network, consequently, the data processing is the main challenge posed by integrating cloud computing with edge computing, it demands researchers to discuss and develop high-performance data processing architectures.

Many researchers provide a hybrid approach between edge computing and cloud computing such as, Du B, et al. (2018)[38] designed Things-edge-cloud computing architecture to enable edge servers to cooperatively work with the cloud and achieve traffic-data as a service. Xu, X. et al. (2017) [14]presented edge Analytics as a service; rule-based analytics model equipped to edge node to prop the management of real-time analytic on edge. Alturki, B et al. (2017)[15] leveraged edge computing to provide low latency and network traffic by using confusion technique to preprocessing data in edge layer and (Fu, J. et al, 2018) [13] proposed data processing schemes which combine the fog computing and cloud computing to improve the quality of service in terms of latency, security, and flexibility and build object-oriented index to enhance data retrieval. They also discuss the impact of distributed services between Fog and cloud Architecture for IoT; four types of architectures have been evaluated with three types of datasets. This evaluation displays the importance of distributed services between fog nodes and cloud computing. In Bittencourt, L. et al, (2018) [39] they discussed the integration of IoT-Fog-Cloud system and provided a review for different aspects such a system are organized managed, and how applications can benefit from it.

Others researchers presented the functionality that edge computing should provide it. Consequently, In (Ai Y et al, 2018) [40], the authors comprehensively present a tutorial on three typical edge computing technologies, namely Mobile Edge Computing, Cloudlets, and Fog computing. In particular, the standardization efforts, principles, architectures, and applications of these three technologies are summarized and compared. From the viewpoint of radio access network, the differences between mobile edge computing and the fog computing are highlighted, and the characteristics of the fog computing-based radio access network are discussed. Finally, open issues and future research directions are identified as well.

Machine learning has an essential role in edge computing thus, Yuuichi Teranishi et al (2017)[41] propose a novel dynamic data flow platform for Internet of Things (IoT)

applications in edge computing environments. To avoid the overloads on network and computational resources that are caused by IoT applications, the proposed platform replicates processes and changes the structure of the data flow dynamically on the distributed computational resources located at network edges and data centers. M. Mahdavinejad et al. (2018) [42] assesses the various machine learning methods that deal with the challenges presented by IoT data by considering smart cities as the main use case. The key contribution of this study is the presentation of the taxonomy of machine learning framework explaining how different techniques are applied to data in order to extract higher level information. The potential and challenges of machine learning for IoT data analytics are also addressed. Sneha Sureddy et al. (2018) [43] present a proposal of Flexible Edge Computing (FEC) architecture as a flexible system to perform edge computing using deep learning in IoT. Combination of these two models that are deep learning and flexible edge computing significantly improve the performance of the system and optimize the task assignment between edge layer and cloud layer.

IV. Data Partitioning Framework

The main functionality of an Edge Computing layer, when integrated with Cloud network, is to improve performance in terms of latency, and network traffic load through processing data. Hence, data processing is the main issue in edge computing in cloud computing real-time application networks such as IoT, CPS, and M2M [44].

In an attempt to overcome the data processing issue, we construct data processing framework over the Edge-cloud network by integrating the function of data partitioning, restructuring, storing and retrieving, to enhance the performance of Edge-cloud network in terms of lower latency level and network traffic. The architecture of the proposed framework consists of three main layers as presented in the flowchart shown in Fig. 3, These layers are edge device layer, edge computing layer, cloud layer.

- Edge device layer: in this layer, several devices are connected and employed for different purposes and these devices are responsible for the aggregation and integration of the data in order to be delivered to the edge node via wireless or wired communication by using a suitable routing algorithm. We also propose that the data sent by edge devices can be used in both control data (i.e. real-time analytics) as well as knowledge extraction data (i.e. historical analytics).
- Edge computing layer: is a communication layer between cloud and end device levels, it is connected to the cloud server through the Internet. This layer receives raw data from the edge device and inputs this data into a uniform preprocessing procedure that issues both cleaning and integration of data. This preprocessed data would form a schema i.e. is stored in a database that facilities data access and search. After that, this schema is partitioned (classified and restructured) based on the features that are selected from user requirements to permit fast access and search on the database as well as to reduce the amount of data transmitted to cloud computing and for saving the bandwidth resources and storage in the cloud layer.
- Cloud layer is employed to perform several processes, such as:

a) Store the transformed data in the cloud data center.

b) Mining tools execute the search operations on data to extract knowledge and new patterns. Data analytic tool are used on historical data for support decision.

c) Analytic tools execute the search operations on data to support decisions.

d) In the training phase, the training classifier and feature selection algorithm are executed at the cloud layer.

e) Queries are updated and added to the queries accumulator accordingly.

f) Control attributes are imposed on the network to provide higher quality of results.

In the following we describe the design issues and constraints on the implementation of the framework and present the data workflow also the network throughput calculation.

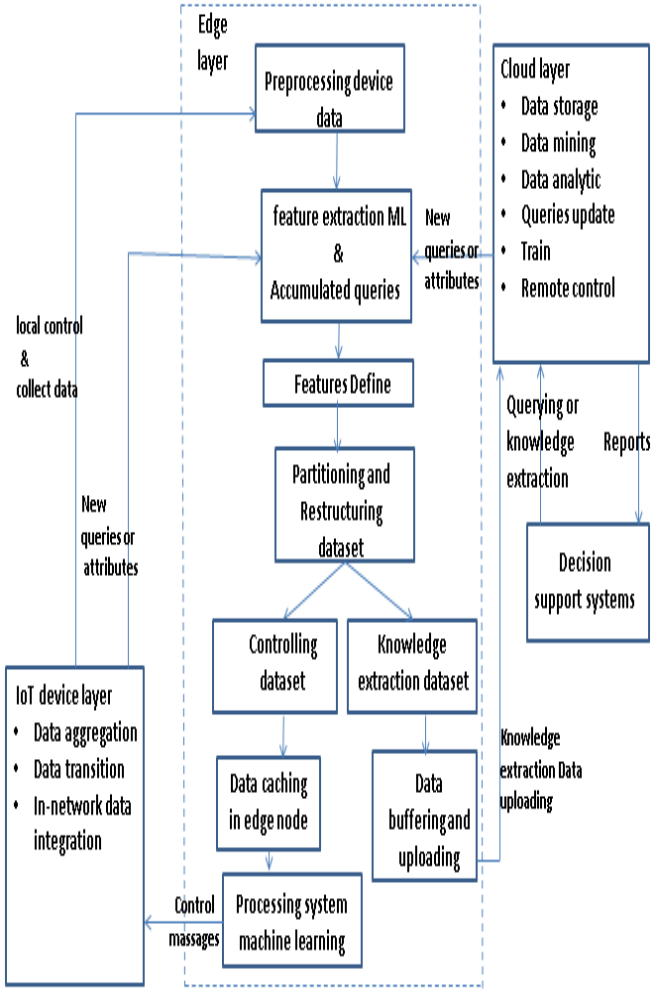

Fig. 3. The Proposed Framework Architecture.

A. Design Issues

Before implementing our framework, there are three important design issues that should be addressed, the requirements analysis to selected features, data classification as control data and knowledge extraction, and the extraction of the subset of features to restructure data schema. These issues are discussed for our proposed framework as follows:

1) Requirements analysis: the requirements analysis issue, as mentioned above, features are selected from user requirements, these requirements can be obtained from a given stream of queries generated over a time period for existing systems or from queries generated by network requirements for a new system. It is the responsibility of the system analysts to extract and detect the features and relationships among them for many large problems.

It is important to mention that user requirements can be past referenced behavior for existing systems; the past referenced behavior is obtained from a given stream of queries generated over a time period. While the requirement for building a new system, are obtained from the users' requirements and network analysis, thus a large sequence of queries can be generated by these requirements, which represents the system referencing behavior.

However, when the requirements queries' size grows too big, the process of analyzing the requirements queries for detecting the features and their frequency is excessively prohibitive. Hence, MapReduce [45] technique was used to overcome this issue.

a) Modify MapReduce algorithm: The traditional MapReduce a programming model [46] was inspired from the Functional Programming model introduced by Google, which uses the Divide and Conquer technique to process large amounts of data. It works in three functions [47]: the map function, the Shuffle Function and the reduce function. The Mapper stage splits the queries into features by whitespaces, and the output of the Mapper is the pair of features with their frequencies known as (key, value) pair, where key is the feature and value is a count of features. Then, in the intermediate phase, Shuffle sorts the (key, value) pair according to the key and sends it to the Reducer. Pairs that have the same key go to the same Reducer.

The Reducer stage collects all (key, value) pairs with the same key and sums the values for each key.

In this paper, we modify the MapReduce algorithm to support our work to find features from user requirements by resetting all values for each key to one as shown in algorithm 1, we reset list (count) number for each feature (attribute) to one, to avoid the duplication of features in one query. The output of this stage is a list that combines the feature Ai with its total frequency where i is the location of the attribute.

2) Data classification: The second issue is the classification of accumulated data into control data and knowledge extraction data based on feature extraction from user requirements. Thus, we used extreme learning classifier with feature selection to classify our data.

Algorithm 1: M-MapReduce algorithm.

```
1.map(file, queries) {
  for each features in queries. Split(){
  output (word, 1);
  }
}
2. Reduce (word, list (count)) {
  Reset all # in list (count) to 1
  Output (word, sum (count));
}
```

3) Features subset extraction: The third issue is extracting the subset of features that have an inherent locality structure, in other words, grouping the features that form temporal locality in the same subset. The features that are requested in the same query, construct temporal locality. The locality structure is discussed in detail in the next subsection. In this dissertation, we proposed using the k-mean algorithm [48][49] [50] which is modified according to our approach, combined with the features selection Wrapper methods which is performed in a new way to cope with our approach as shown in Fig. 4 [51] [52]. This combination provides a machine learning subset system that is used to find a subset of features.

a) Machine Learning Subset System: In this dissertation a machine learning subset system was used in establishing a subset of features; it groups features into subsets according to their frequency.

The traditional k-means algorithm consists of two stages. The first stage adjusts k based on the number of groups [49][50]. The second stage detects initial centroid randomly from the dataset for each group. In this dissertation, we used k-means with modifying its stages (M_k-means), where the initial centroid is determined based on a features frequency (i.e., taking the higher and lower frequency) and the number of groups K determines based on used a wrapper feature selection method. However, we illustrated the steps of this system as follows:

The M K-means algorithm as shown in Algorithm 2 in uses iterative refinement to produce a final result. The algorithm inputs are the number of groups K and the data set. The data set is a collection of features (data attributes) and frequency for each feature. The algorithm starts with two groups; the first group takes the high value of a feature frequency as a centroid while the second group takes the lower value of a feature frequency as a centroid. The initial value of group centroid(fai) was defined to avoid the oscillation with iterative refinement to produce a final result.

Modify k-mean algorithm: The K means algorithm is used to establish a subset of features, it groups features into subsets according to their frequency [49][50]. Traditional K means

algorithm consists of two stages. The first stage adjusts k based on the number of groups. The second stage detects initial centroid randomly from the dataset for each group. In this paper, we used k-means with modifying the second stage (M_k-means), where the initial centroid is determined based on feature's frequency (i.e. taking the higher and lower frequency) as shown in algorithm 2.

Algorithm 3.2: M_ K-means Algorithm

Input:

F = {f1a1, fa2,.....,fnai} // fai is a feature frequency ,i is a feature # and n is the number of feature.

k // Number of desired subset

Output:

A set of k subset.

Steps:

Phase 1: Determine the initial centroids of the subset (group) by taking high and low frequency (fnai).

Phase 2: Calculate new mean for each cluster;

Until convergence criteria is met.

The algorithm iterates between three steps and iterates among these steps until a stopping criterion is met. These steps illustrated in detail as follows:

Data Assignment Step 1

In this step, each data point is assigned to its nearest centroid, based on its frequency fi. More formally, if ci is the centroid in cluster C, then each data point x is assigned to a cluster based on the following:

$$dif\ (fi, ci) \quad (1)$$

Where $dif\ (\cdot)$ is the difference between the feature's frequency and cluster's centroid. Let the number of features assignments for each ith cluster centroid is S_i.

Centroids Update Step 2:

In this step, the centroids are recomputed. This is done by taking the mean of all frequency of features to that centroid's cluster.

$$Ci = \frac{1}{|Si|}\sum_{fi \in Si} fi \quad (2)$$

Choosing K Step 3:

In the previous two steps, the clusters and data set labels were found for a particular pre-chosen k. In this step, we discuss how wrapper feature selection is used in a new method to choose the number of clusters K.

Some of the techniques that are commonly used to find k are: cross-validation, information criteria, the information-theoretic jump method, the silhouette method, and the G-means algorithm. In this dissertation we proposed a new method of wrapper feature selection to find the value of k.

The wrapper is a feature selection method, that evaluates feature subsets by the quality of the performance on a modeling algorithm, which is taken as a black box evaluation [51] [52]. In our approach, the wrapper will evaluate subsets based on the performance of executing queries on the new tables that are built on the feature subset.

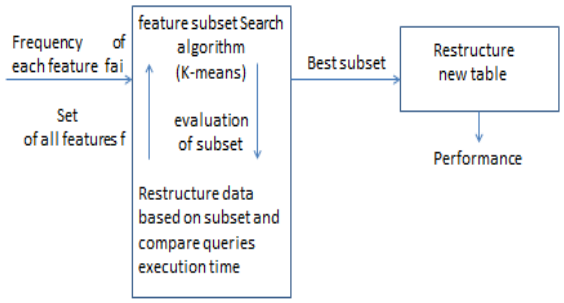

Fig. 4. Our Approach Wrapper Feature Selection.

The evaluation is repeated for each subset and the subset generation is dependent on the k-means algorithm as shown in Fig. 4.

B. Data Scheme Partitioning Workflow

The Preprocessing data scheme was partitioned by some processes as shown in the flowchart in Fig. 3 which is assigned by a dotted line. These processes and their design issues can be described in the following two parts. First, requirement analysis (feature selection) and Second, partitioning data schema.

1) Requirement analysis (feature selection): This part is responsible for detecting the locality structure of control data and knowledge extraction data; it relies heavily on user requirements or query stream analysis.

The objective of this part is to group attributes, which are likely to be referenced together , in a partition or table. The more frequently the attributes are referenced together, the more likely for them to belong to one locality structure. for example, in a monitaring system in a factory, the monitoring devices share the same identifier, also in medical systems the diseases may be chronic or not chronic.

Query stream analysis is used to find the frequency of a given query execution and the frequency of attributes requested within the same query. Furthermore, query analysis is used to detect the adjacency of attributes, where adjacency is defined as the attributes referenced within the same query. Attribute Ai is said to be adjacent to Aj: AJ (Ai,Aj) if Ai and Aj are requested within the same query Q_k. By default, adjacent attributes form a temporal locality, since they are referenced within the same query and their access to the data storage falls within the same time period. Note that adjacent attributes Ai and Aj may or may not be stored within the same virtual page or in close space proximity.

For example, assume that query Qk is used to select attribute Ai and attribute Aj from table T1. Thus, the attributes Ai and Aj are adjacent within the query Q. Ai and Aj certainly form a temporal locality, in the sense that both are referenced within the same period. The system will access both A_i and A_j and return their values as requested. However, A_i and A_j may be very well located in different memory regions, or more

precisely could be located in two different pages in the virtual space. If Q_k is referenced n times, then it is only reasonable to have A_i and A_j stored in close proximity, e.g., in the same page or the same page block, such that when the page or page block is transferred from virtual storage to main memory or cache, then the requested items A_i and A_j will have been located in main memory already.

For illustration purposes, assume that the data table has 10^9 which is ≈ (2^{30}) records, and attributes A_i and A_j are both of type real with 8 bytes each. This means that each column representing attributes A_i and A_j is 2^{33} bytes. Assume further that a page size (using virtual memory paging representation), is 1024 Kbytes (2^{20} bytes). Thus the number of pages representing each attribute is given by $2^{33}/2^{20} = 2^{13}$ pages (≈ 8 GBytes). If attributes A_i and A_j are requested as adjacent attributes frequently, i.e., they appear in the same query or in different queries N times, where N is relatively large, then it is worthwhile to convert the temporal locality of A_i and A_j to spatial locality, in the sense that A_i and A_j should be adjacent in space allocation scheme. Fig. 5 shows a temporal and spatial locality example.

Consequently, data are restructured based on adjacency of attributes where the unrelated attributes are removed. Furthermore, these steps' outline is illustrated in Algorithm 3.

Moreover, the attributes that have the highest frequency value (Ati) will be collected in the same new table.

2) Partitioning data scheme: This part is responsible for partitioning the dataset based on features that are selected in the previous part. Partitioning aims to classify the data in order to remove unrelated data to reduce the size of data. As well as grouping a set of attributes in a subset, where the subset will be used to create a new table of data with the attributes given in the subset. In essence, the process of partitioning leads to the auto reconstructing of the original schema, which was originally used to represent the data. This part includes the following steps.

a) Classifying the data as controlling data and knowledge extraction data using feature selection in part 1. Machine learning classification algorithm (classifier) with feature selection is used to classify the data instance, this classifier learning in the cloud layer is based on features that were selected from user requirements. Then it is used to classify data in the edge layer.

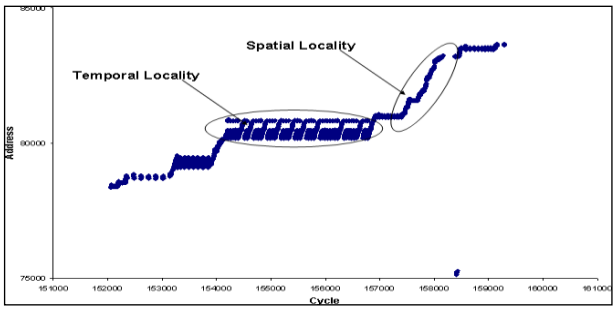

Fig. 5. Temporal and Spatial Locality Example.

ALGORITHM 3: GENERATE ATTRIBUTE ADJACENCY

1. Generate and collect a set of queries {Q1}{q1}&{q2}. // Requirements analysis (Requirements engineering).
2. Analyze queries and extract the following parameters:
   a. For each query (q), find the frequency of its execution f(q).
   b. Build adjacency matrix for all attributes contained within the set of queries {Q}.
      i. Two attributes Ai and Aj are said to be adjacent if both attributes are referenced within the same query.
      ii. The number of attributes Ai that is shared between the query Qi and Qi, is given by the similarity S (q).
      iii. The number of appearance Ai within different queries is given by fai.
      iv. The total frequency of an attribute Ati = fqi * fai
3. Build adjacency list for features selected from q1 contained within the set of queries {Q1}.
4. Build adjacency list for features select from q2 contained within the set of queries {Q2}.

b) For the control data, create a new table for each subset in the set Si, these new tables are stored in the edge layer for real-time analysis. For this step, we develop algorithm 3.

c) Creating a new table for the knowledge extraction data is suitable for the data mining technique using feature selection from q2 and uploading to the cloud layer for storing and analysis.

These steps are performed on our framework by applying algorithms 4.

C. Network Throughput

Without loss of generality and accuracy, we will ignore the complexity of algorithms 3.3 and 3.4, since they will be performed one time whenever the structuring of the DB schema is required. Henceforth, we are not concerned with the complexity of the algorithms; rather our concern is with the complexity of the data access time as well as data transmission time before and after partitioning.

Algorithm 4 Partitioning Data Schema

1) Create list A of attribute Ai and frequency n (Ai ,n).
2) Sort Ai in list A according to largest frequencyn.
3) Using the wrapper feature selection method with the execution time of query as an evaluation strategy.
   a) Running k-means algorithm to Grouping attribute in a subset Si = {s1, s2…sn}.
   b) Evaluate Si using the evaluation function
   c) While (stopping function not met)
   d) Fined new Si
   e) Evaluate function
   f) End while
   g) Output: the best Si
4) If Have new query go to step 1
5) Terminate Algorithm.

Assume that the DB tables are stored in a column wise schema, where the data belonging to one attribute are stored sequentially. Then to retrieve one record in a table, it is required to scan all attributes in the record. Given that the number of attributes in a table is N; this leads us to consider the space occupied by the table as a major cost of the system. When multiplied by the overall time required executing a query, or the time required transmitting this table the complexity naturally lends itself to the space time product as the main tuning parameter.

It has been shown that the space-time product of a task or a set of tasks is inversely proportional to the throughput of the system (Denning & Buzen, 1978). In other words, if we want to maximize the throughput (X), which is defined as the number of tasks performed within a time period(T), then we have to minimize the space-time cost (Y), i.e., the total space (S) consumed by the tasks within the same time period (T). This implies the following relationship.

$$X \approx 1/Y, \text{ where } Y = S*T \quad (3)$$

1) Data retrieval complexity: In term of data access time and reduce latency**,** for a given query (Q_i), the space consumed during the query execution equals the space of the tables referenced by Q_i. For example, assume that Q_i has the following structure:

$Q_i \rightarrow (\{A_i\})$ from Tables $\{T_k\}$

Where a set of attributes A_i , i =1, .. n are referenced in tables T_k, k = 1, … m

The space occupied by Q_i is thus given by Equation 2

$$S(Q_i) = \Sigma S(T_k) , k = 1,\dots m \quad (4)$$

And the time required to execute the queries is given by:

$$T = t*An \quad (5)$$

Where A_n is the number of attributes referenced in Q_i and t is the time required to process each attribute.

Thus the space time cost of a query Q_i is given by:

$$Y = T_i*S \quad (6)$$

If the frequency of Q_i is *f* then the total space time cost for Q_i is given by:

$$Y = f* T_i*Si \quad (7)$$

2) Data transmission complexity: While, in term of data transmission time, for transmitted a table Tk, the transmission time is related to the space of the table. For example assume that the table T has Ai attributes where Ai , i =1, n are referenced in tables and Rj records, j= 1…..m

The space occupied by T is given by equation 6

$$S(T)= \Sigma Rj\ Ai \quad (8)$$

And time required to transmit the table Tk is Tt

$$Tt= t* An \quad (9)$$

where A_n is the number of attributes referenced in Tk and t is the time required to transmit each attribute.

Thus, the space time cost of a table Tk is given by:

$$Y = T_t*S(T) \quad (10)$$

The objective of the proposed method is to maximize the throughput of the system by minimizing the space-time product for the data access (i.e., reduce latency), and data transmission (i.e., saving bandwidth). Therefore, the throughput is increased with decreasing space (S).

V. Analysis and Results

To verify the performance of the proposed framework of data partitioning, we assumed there is an edge cloud network deployed in a chronic kidney disease center to monitor the patient's status. We experimented our proposed framework on this network and compared it to the same network, without our framework. We compared between these networks in term of data transmission amount, storage space and queries execution time (data retrieval efficiency). The proposed framework is experimented with different queries and different sizes of data. The utilized dataset, experiment environment, queries generation, and the results with their discussion are given as follows.

This verification takes on two aspects:

- Experimental Verification which begins by defining the experiment processes, dataset used, environment, and query generation, followed by a discussion of the results in Section A.
- Theoretical Verification which covers the theoretical verification for the proposed framework on this network and is discussed in Section B.

A. Experimental Verification

This section describes, in detail, the experiment performed to verify our framework.

1) Processes

a) Requirements analysis with M-MapReduce.

b) Data classification using ELM classifier.

c) Data restructuring using our subsystem machine learning.

d) Generation of queries executed against the new data schema.

2) Dataset: In this research, we use data from the UCI repository. The data collected from the Apollo Hospital, India by b. Jerlin Rubini. The number of instances in the dataset are 400 instances with 25 attributes (including attribute classes), where 250 instances include those who have chronic kidney disease (ckd) and the remaining 150 who did not have chronic kidney disease (notckd) as in Table I.

3) Environment: In this experiment, our edge server consisted of a laptop computer with a 2.6 GHz Intel Core processor, Window 7 operating system and 4 GB of RAM. As discussed previously, the edge server handles running the network and analyses the pre-processed data. Our cloud server was a desktop computer with a 3.6 GHz Intel Core processor, Windows 7 operating system and 8 GB of RAM.

The edge server connects to the cloud server through the Internet and pre-processes the raw data. We used MATLAB® R2018b extreme learning machine (ELM) libraries for the classification methods and a k-means function for the clustering method.

The MapReduce algorithm is used to identify suitable features. We first taught the classifier in the cloud server, and then applied steps one and two in the previous section on our data to define the features that are used in restructuring the data. We also created the database structures necessary for the new tables that are created.

4) Edge layer query generation and cloud layer feature selection: Based on the users' requirements, we generated random queries to apply our approach. Specifically, from the users' requirements in the edge layer, 120 queries are generated randomly. In Table II, the 10 edge layer generation queries that had the highest frequency are listed. From the expert and user requirements in the cloud layer, we can extract and select relevant feature subsets. These subsets, shown in Table III, are used to restructure the data passed to the cloud layer.

Based on the requirements noted above, as well as from network requirements, we selected the features that classified the data into two groups: CKD, and not CKD. The ELM, using a wrapper method for feature selection, was used to classify this data [53][54].

We applied Algorithm 3 on the generated queries, to select the features to use in restructuring the data. Table IV (an adjacency matrix for all the attributes within the set of queries and a list for each attribute with its frequency), Table V and Table VI illustrate the features for the controlling data and knowledge extraction data, respectively. After the feature selection, we performed the following two steps.

TABLE. I. ATTRIBUTES OF CKD DATASET

No.	Code	Parameter	Value
1	Age	Age	In the year
2	Blood pressure	Blood pressure	In mm/Hg
3	Specific gravity	Specific gravity	(1.005,1.010,1.015,1.020,1.025)
4	Albumin	Albumin	(0,1,2,3,4,5)
5	Sugar	Sugar	(0,1,2,3,4,5)
6	Red blood cells	Red blood cells	Normal / Abnormal
7	Pus cell	Pus cell	Normal / Abnormal
8	Pus cell clumps	Pus cell clumps	Present / NotPresent
9	ba	Bacteria	Present / NotPresent
10	bgr	Blood glucose random	In mgs/dl
11	bu	Blood urea	In mgs/dl
12	sc	Serum creatinine	In mgs/dl
13	sod	Sodium	In mEq/L
14	pot	Potassium	In mE1/L
15	hemo	Hemoglobin	In gms
16	pcv	Packed cell volume	In %
17	wc	White blood cell count	In cells/cumm
18	rc	Red blood cell count	In millions/cmm
19	htn	Hypertension	Yes / No
20	dm	Diabetes mellitus	Yes / No
21	cad	Coronary artery disease	Yes / No
22	appet	Appetite	Good / Poor
23	pe	Pedal edema	Yes / No
24	ane	Anemia	Yes / No
25	Class	Class	Ckd, notckd

TABLE. II. EDGE LAYER GENERATION QUERIES

Q1→ bp, sg, bu, sod, pot, hemo, ckd	Q7→ bp, bu, sod, hemo, ckd
Q2→ bp, sg, bu, sc, sod, hemo, ckd	Q8→ bp, sg, pot, hemo, ckd
Q3→ bp, sg, bu, sc, sod, ckd	Q9→ bp, sg, bu, sc, pot, ckd
Q4→ bp, sg, bu, sod, hemo, ckd	Q10→ sg, bu,sod
Q5→ bp, bu, sc, sod, pot, hemo, ckd	Q11→ pb, sg, bu, sc, sod, pot, hemo, ckd
Q6→ bp, sc, sod, pot, ckd	----

TABLE. III. CLOUD LAYER GENERATION QUERIES

Q1→Hypertension, CKD
Q2→Diabetes mellitus, Hypertension, CKD
Q3→Diabetes mellitus, CKD

TABLE. IV. FEATURES OF CONTROLLING DATA (ADJACENCY MATRIX)

Attr / Q	Age	Bp	sg	al	su	rbc	Pc	pcc	ba	bgr	Bu	sc	sod	pot	Hemo	pcv	Wc	Rc	htn	Dm	cad	appet	pe	ane	Fq
Q1	0	1	1	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	4
Q2	0	1	1	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	4
Q3	0	1	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	3
Q4	0	1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	4
Q5	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	3
Q6	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	3
Q7	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	4
Q8	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	3
Q9	0	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	4
Q10	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	3
Q11	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	48
Fai for 120 q	**5**	**93**	**97**	**5**	**16**	**17**	**5**	**5**	**16**	**5**	**103**	**94**	**96**	**98**	**96**	**18**	**20**	**8**	**17**	**15**	**16**	**15**	**19**	**19**	

TABLE. V. FEATURE OF KNOWLEDGE EXTRACTION DATA

Hypertension
Diabetes mellitus
Class

TABLE. VI. KNOWLEDGE EXTRACTION DATA SCHEMA

Hypertension	Diabetes mellitus

TABLE. VII. CONTROLLING DATA SCHEMA

Blood	Specific-Gravity	Blood-Urea	Serum Creatinine	Sodium	Potassium	Hemoglobin

Firstly, we restructured the knowledge extraction data in one table by using the features in Table V and buffered it for transmission to the cloud. This table schema is shown in Table VI.

Secondly, we used Algorithm 4 on Table IV to restructure new tables for the controlling data. The k-means algorithm was applied twice, once with k = 2 and again with k = 3. A superior subset was achieved with k = 3. The algorithm provided three subsets, and two of these subsets contained features with low frequency. Consequently, we built only one table using the subset that contained features with high frequency, and that represented the controlling data which is then stored in the edge layer. The schema of this new table is shown in Table VII.

5) Results: As previously mentioned, our framework is tested for data transmission time, storage space and queries execution time (i.e., data retrieval efficiency).

a) Data Transmission Time and Storage Space: As mentioned above, from user requirements and network requirements, the pre-processing data were separated into two classes: CKD, and not CKD. Only the CKD class is relevant. Thus, the amount of the processing data was reduced. This conclusion is supported by Fig. 6 which shows both the original data and relevant classification data plots. The classification data clearly required a lower amount of storage and analysis.

Fig. 7(a) compares the amount of data transmitted over the network from the edge layer to the cloud layer for storage. The two bars represent the two datasets: the red bar for original dataset before any partitioning, and the blue bar for the dataset after partitioning (classification and restructuring, as shown in Table VI). Clearly the volume of data transmitted over the network for the original dataset is extremely high relative to the volume transmitted over the network for the dataset after partitioning. Fig. 7(b) portrays the corresponding transmission times of the two datasets. This result confirms that a network with our framework can reduce network resource requirements (i.e., bandwidth and storage space) and, as a direct consequence, reduce network traffic because of the increased rate of data transmission.

In Fig. 7(a) the plotted bar of stored value indicates the amount of stored data is 92 KB for the network for the original dataset and only 6.95 KB for our partitioned dataset. This represents a 92.4% reduction in storage space required. Similarly, in Fig. 7(b), the plotted bar of stored value indicates the amount of transmission time is 0.007376 S for the original dataset and only 0.000556 S our partitioned dataset. This represents a 92.5% reduction in time required for transmission.

b) Query Execution Time (Data Retrieval Efficiency): In this section, we compare the execution time of the 11 queries in Table II with different size datasets, and examine the outcome for two additional queries in Table VIII on both the original dataset and our new schema dataset(partitioning dataset) (see Table VII).

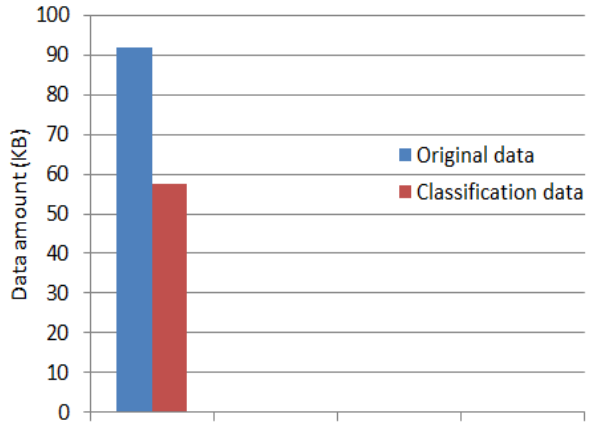

Fig. 6. Data Classification Amount.

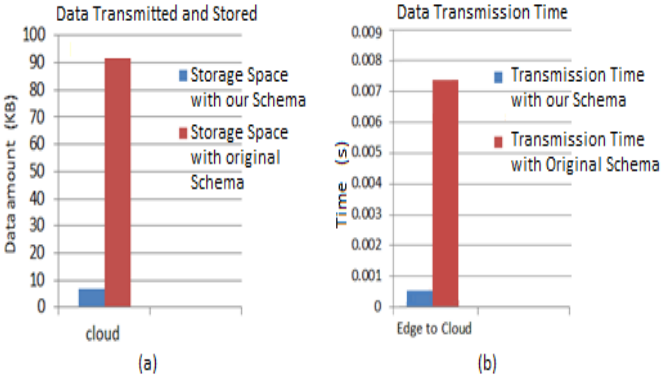

Fig. 7. (a) Data Storage Space and (b) Transmission Time.

Table VIII shows the execution time of the 11 queries in both our schema dataset and the original dataset. The results show significant differences in the execution times between the two datasets. Obviously, our schema dataset requires a shorter execution time than the original dataset because it has a smaller size as well as greater spatial locality of reference. In other words, the network with our framework has low latency and high performance. These differences can be shown by analyzing the bar chart in Fig. 8. Furthermore the average execution time for these 11 queries was decreased by 79.841%.

An interesting observation is the execution time of the Q5 which is low in both datasets because the temporal locality of the query schema is virtually identical to the spatial locality of both types of data schema.

Table IX shows the execution times produced using a relatively large dataset. By comparing Table IX with Table VIII, we observe that the proposed framework consistently has the shorter execution time despite the change in data size. Fig. 9 illustrates graphically that the difference between execution times persists in spite of the change in the dataset size.

We conclude that changes in dataset size do not adversely affect the results, meaning our proposed framework can achieve its objectives in different networks with different dataset sizes.

Fig. 10 compares the execution times of the two additional queries in Table X against each of the two sizes of dataset. Again, the proposed framework has a consistently shorter execution time compared to the original system.

TABLE. VIII. QUERY EXECUTION TIME FOR NORMAL SIZE DATASET

Query	Execution Time with Our Schema	Execution Time with Original Data
Q1	0.0025	0.0027
Q2	0.0020	0.0025
Q3	0.0015	0.0025
Q4	0.0023	0.0025
Q5	0.0005	0.0010
Q6	0.0022	0.0025
Q7	0.0020	0.0025
Q8	0.0015	0.0025
Q9	0.0020	0.0025
Q10	0.0012	0.0025
Q11	0.0025	0.0026

Fig. 8. Query Execution Times.

TABLE. IX. QUERY EXECUTION TIMES FOR LARGE SIZE DATASET

Query	Execution Time with Our Schema	Execution Time with Original Data
Q1	0.0040003	0.0080003
Q2	0.0050003	0.0070003
Q3	0.0040003	0.0060005
Q4	0.0030003	0.0040003
Q5	0.0022000	0.0030003
Q6	0.0025000	0.0030000
Q7	0.0030000	0.0040000
Q8	0.0030003	0.0035000
Q9	0.0040001	0.0045005
Q10	0.0025005	0.0030000
Q11	0.0040000	0.0060005

Fig. 9. Query Execution Times (Large Dataset).

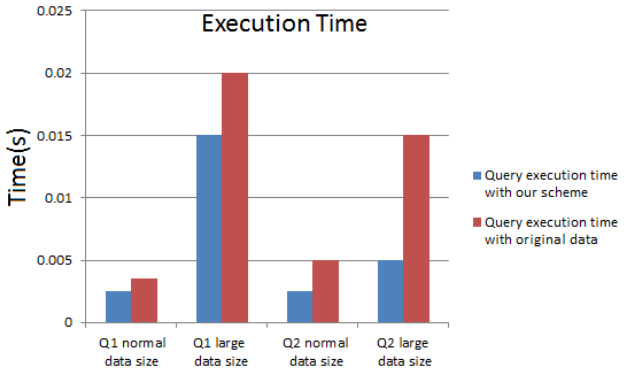

Fig. 10. Queries Execution Time.

TABLE. X. TWO SUGGESTED QUERIES

Q1→	Blood, Specific_Gravity, Blood_Urea, Sodium, Potassium, Hemoglobin where Specific_Gravity like "1"
Q2→	Blood group by Hemoglobin

B. Theoretical Analysis and Verification

This section describes the use of the throughput calculation functions in Section IV to theoretically evaluate our framework in the proposed system. This calculation involves two elements, provided by Equations 5 and 7: the first element is the measurement of data access time, performed by calculating the space-time for demand query. The second element is to calculate the data transmission volume, performed by computing the space-time product for data moving through the network.

1) Dataset access time: As mention before, the original dataset of our system has 400 instances and 25 attributes. Considering our approach, this dataset after classification and restructuring to identify a controlling dataset, a new dataset was built. This new dataset has 250 instances and 7 attributes as shown in Table VII.

Table XI displays the results of performing equation 5 (i.e., Y = f * Ti * Si) on the queries in Table II, to calculate the space-time product of the proposed network with and without our framework.

As shown in Table XI, where the first column identifies the query, the second column shows the space-time product for the original dataset, and the third column displays the space-time product for the new dataset. Our proposed framework consistently showed a more favorable result. This conclusion is further reinforced when examining the bar chart shown in Fig. 8 showing execution times for both the original dataset and new dataset. Indeed, a lower latency is invariably achieved by our approach. The average execution time was reduced by 98%.

The storage of data is critical to the process of retrieving data in real-time. By extension, it necessarily has a material impact on network latency. The result in Table IX shows that best throughput is achieved with our approach when the data storage is considered and takes into account the fact that data which exhibits good locality structure can be accessed more easily than data with poor locality.

2) Data transmission time: The original dataset of our system has 400 instances and 25 attributes. Using our approach, this dataset, after classification and restructuring to establish a knowledge extraction dataset, a new dataset was built containing 250 instances and only two attributes.

Thus, the data transmission space-time before partitioning data may be expressed by the following equation:

$$Y = T * S(T)$$

$$Y = 400 * 25 * t = 10000\ t$$

The data transmission space-time after partitioning data is expressed as follows:

$$Y = 250 * 2 * t = 500\ t$$

According to the results of the space-time product of transmission data, our framework proves a better approach for analyzing the dataset. This is confirmed by observing the chart in Fig. 6 which shows the amount of data transmitted to the cloud layer and stored there as well the time required to transmit the data. The space-time product of transmission data rate was reduced by 95%

TABLE. XI. CALCULATION OF SPACE-TIME PRODUCT

Query Number	Space-Time Product of Original Dataset	Space-Time Product of New Dataset
Q1	4 * 6 * 400 * 25 =240 000	4 * 6 * 250 * 7 = 42 000
Q2	4 * 6 * 400 * 25 =240 000	4 * 6 * 250 * 7 = 42 000
Q3	3 * 5 * 400 * 25 =150 000	3 * 5 * 250 * 7 = 26 250
Q4	4 * 5 * 400 * 25 =200 000	4 * 5 * 250 * 7 = 35 000
Q5	3 * 6 * 400 * 25 =180 000	3 * 6 * 250 * 7 = 31 500
Q6	3 * 4 * 400 * 25 =120 000	3 * 4 * 250 * 7 = 21 000
Q7	7 * 4 * 400 * 25 =280 000	7 * 4 * 250 * 7 = 49 000
Q8	3 * 4 * 400 * 25 =120 000	3 * 4 * 250 * 7 = 21 000
Q9	4 * 5 * 400 * 25 =200 000	4 * 5 * 250 * 7 = 35 000
Q10	3 * 3 * 400 * 25 =90 000	3 * 3 * 250 * 7 = 15 750
Q11	48 * 7 * 400 * 25 =3 360 000	48 * 7 * 250 * 7 = 588 000

Network throughput quantifies the amount of data, during a defined time interval that a network can send or receive. Throughput must take into account the entire network overhead as well as contention on the transmission links. Multiple data flows on a link will each use some percentage of the overall bandwidth, thereby reducing the total throughput of each. It follows that increasing data flow causes an increase in both network traffic and contention on the links.

Bandwidth is the number of bits per second that a link can send or receive, including all flows. Data rate (or data transfer rate) is the volume of data transferred through a connection within one second. The data rate cannot exceed the bandwidth of the connection; data rate is closer to bandwidth. Thus, the reducing the required data rate reduced traffic overhead on the network; in other words, a reduced data rate leads to a reduce bandwidth requirement.

VI. Conclusion

In this paper, efficient data storage and retrieval frameworks are proposed based on both the edge computing and cloud computing techniques. The main challenges in terms of data partitioning and requirements analysis are summarized, and appropriate solutions are also provided. Specifically, the data partitioning (classification and restructuring) framework is provided to support low latency and save network bandwidth based on MapReduce algorithms, wrapper feature selection method and a machine learning proposed subsystem, in addition, a new algorithm for restructuring the data was proposed.

The components of the proposed framework flowchart are discussed; data partitioning (knowledge extraction and control data) and requirements generation are presented from a wider view. The functionalities of each point of the proposed framework are displayed in detail.

The framework is verified on a case study and approved its efficiency that is evaluated using Data time, storage space, and data retrieval time.

For future work, other data restructuring algorithms can be used and other types of requirements analysis models can be explored.

References

[1] K.-D. Kang, D. S. Menasche, G. Küçük, T. Zhu, and P. Yi, “Edge computing in the Internet of Things,” Int. J. Distrib. Sens. Networks, vol. 13, no. 9, p. 155014771773244, 2017.

[2] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, “Fog/Edge Computing-based IoT (FECIoT): Architecture, Applications, and Research Issues,” IEEE Internet Things J., no. October, 2018.

[3] D. Boswarthick, O. Elloumi, and O. Hersent, M2M communications: a systems approach. John Wiley & Sons, 2012.

[4] P. C. Evans and M. Annunziata, “Industrial internet: pushing the boundaries of minds and machines, 2012,” Available(accessed 4.10. 2016) https//www. ge. com/docs/chapters/Industrial_Internet. pdf, 2015.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of things for smart cities,” IEEE Internet Things J., vol. 1, no. 1, pp. 22–32, 2014.

[6] L. Qian, Z. Luo, Y. Du, and L. Guo, “Cloud computing: An overview,” in IEEE International Conference on Cloud Computing, 2009, pp. 626–631.

[7] N. Takahashi, H. Tanaka, and R. Kawamura, “Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing,” in 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, 2015, pp. 233–234.

[8] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, “A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications,” IEEE Internet Things J., vol. 4, no. 5, pp. 1125–1142, 2017.

[9] “□,” 2016.

[10] Y. Zhang, H. Wang, J. Zhao, and B. An, “SeCEE: Edge Environment Data Sharing and Processing Framework with Service Composition,” in International Conference of Pioneering Computer Scientists, Engineers and Educators, 2018, pp. 33–47.

[11] B. Du, R. Huang, Z. Xie, J. Ma, and W. Lv, “KID Model-Driven Things-Edge-Cloud Computing Paradigm for Traffic Data as a Service,” IEEE Netw., vol. 32, no. 1, pp. 34–41, 2018.

[12] H. Li, K. Ota, and M. Dong, “Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing,” IEEE Netw., vol. 32, no. 1, pp. 96–101, 2018.

[13] J. S. Fu, Y. Liu, H. C. Chao, B. K. Bhargava, and Z. J. Zhang, “Secure Data Storage and Searching for Industrial IoT by Integrating Fog Computing and Cloud Computing,” IEEE Trans. Ind. Informatics, vol. 14, no. 10, pp. 4519–4528, Oct. 2018.

[14] X. Xu, S. Huang, L. Feagan, Y. Chen, Y. Qiu, and Y. Wang, “EAaaS: Edge Analytics as a Service,” Proc. - 2017 IEEE 24th Int. Conf. Web Serv. ICWS 2017, no. 1, pp. 349–356, 2017.

[15] B. Alturki, S. Reiff-Marganiec, and C. Perera, “A hybrid approach for data analytics for internet of things,” in Proceedings of the Seventh International Conference on the Internet of Things, 2017, p. 7.

[16] J. Li and H. Liu, “Challenges of feature selection for big data analytics,” IEEE Intell. Syst., vol. 32, no. 2, pp. 9–15, 2017.

[17] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, Feature extraction: foundations and applications, vol. 207. Springer, 2008.

[18] Z. Hu, Y. Bao, T. Xiong, and R. Chiong, “Hybrid filter–wrapper feature selection for short-term load forecasting,” Eng. Appl. Artif. Intell., vol. 40, pp. 17–27, 2015.

[19] K. Pohl, Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated, 2010.

[20] R. Bunt and C. Williamson, Temporal and spatial locality: A time and a place for everything. na, 2003.

[21] P. J. Denning and J. P. Buzen, “The operational analysis of queueing network models,” ACM Comput. Surv, vol.10, no.3, pp. 225–261, 1978.

[22] A. S. Levey and J. Coresh, “Chronic kidney disease,” Lancet, vol. 379, no. 9811, pp. 165–180, 2012.

[23] B. Ravandi and I. Papapanagiotou, “A self-learning scheduling in cloud software defined block storage,” in 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), 2017, pp. 415–422.

[24] B. Zhang et al., “The cloud is not enough: Saving iot from the cloud,” in 7th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 15), 2015.

[25] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, “The future of mobile cloud computing: Integrating cloudlets and mobile edge computing,” in 2016 23rd International conference on telecommunications (ICT), 2016, pp. 1–5.

[26] M. Satyanarayanan, “The emergence of edge computing,” Computer (Long. Beach. Calif)., vol. 50, no. 1, pp. 30–39, 2017.

[27] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The case for vm-based cloudlets in mobile computing,” IEEE pervasive Comput., no. 4, pp. 14–23, 2009.

[28] C. M. S. Magurawalage, K. Yang, L. Hu, and J. Zhang, “Energy-efficient and network-aware offloading algorithm for mobile cloud computing,” Comput. Networks, vol. 74, pp. 22–33, 2014.

[29] K. Dolui and S. K. Datta, “Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing,” in 2017 Global Internet of Things Summit (GIoTS), 2017, pp. 1–6.

[30] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, “Fog computing: Focusing on mobile users at the edge,” arXiv Prepr. arXiv1502.01815, 2015.

[31] M. Taneja and A. Davy, "Resource aware placement of data analytics platform in fog computing," Procedia Comput. Sci., vol. 97, pp. 153–156, 2016.

[32] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012, pp. 13–16.

[33] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The Role of Edge Computing in Internet of Things," IEEE Commun. Mag., vol. 56, no. 11, pp. 110–115, 2018.

[34] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," IEEE Trans. Wirel. Commun., vol. 18, no. 1, pp. 695–708, 2018.

[35] P. Wang, C. Yao, Z. Zheng, G. Sun, and L. Song, "Joint Task Assignment, Transmission, and Computing Resource Allocation in Multilayer Mobile Edge Computing Systems," IEEE Internet Things J., vol. 6, no. 2, pp. 2872–2884, 2018.

[36] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," IEEE Internet Things J., vol. 6, no. 2, pp. 3512–3524, 2018.

[37] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An edge-computing based architecture for mobile augmented reality," IEEE Netw., 2019.

[38] B. Du, R. Huang, Z. Xie, J. Ma, and W. Lv, "KID model-driven things-edge-cloud computing paradigm for traffic data as a service," IEEE Netw., vol. 32, no. 1, pp. 34–41, 2018.

[39] L. F. Bittencourt et al., "The Internet of Things, Fog and Cloud Continuum: Integration and Challenges," 2018.

[40] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: a primer," Digit. Commun. Networks, vol. 4, no. 2, pp. 77–86, 2018.

[41] Y. Teranishi, T. Kimata, H. Yamanaka, E. Kawai, and H. Harai, "Dynamic Data Flow Processing in Edge Computing Environments," Proc. - Int. Comput. Softw. Appl. Conf., vol. 1, pp. 935–944, 2017.

[42] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," Digit. Commun. Networks, vol. 4, no. 3, pp. 161–175, 2018.

[43] S. Sureddy, K. Rashmi, R. Gayathri, and A. S. Nadhan, "Flexible Deep Learning in Edge Computing for IoT," Int. J. Pure Appl. Math., vol. 119, no. 10, pp. 531–543, 2018.

[44] V. C.P and D. A. A. Chikkamannur, "IOT future in Edge Computing," Int. J. Adv. Eng. Res. Sci., vol. 3, no. 12, pp. 148–154, 2016.

[45] R. Grossman and Y. Gu, "Data mining using high performance data clouds: experimental studies using sector and sphere," in Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, pp. 920–927.

[46] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[47] A. F. Gates et al., "Building a high-level dataflow system on top of Map-Reduce: the Pig experience," Proc. VLDB Endow., vol. 2, no. 2, pp. 1414–1425, 2009.

[48] Y. Kim, W. N. Street, and F. Menczer, "Evolutionary model selection in unsupervised learning," Intell. data Anal., vol. 6, no. 6, pp. 531–556, 2002.

[49] K. A. A. Nazeer and M. P. Sebastian, "Improving the Accuracy and Efficiency of the k-means Clustering Algorithm," in Proceedings of the world congress on engineering, 2009, vol. 1, pp. 1–3.

[50] C. Slamet, A. Rahman, M. A. Ramdhani, and W. Darmalaksana, "Clustering the Verses of the Holy Qur'an using K-Means Algorithm," Asian J. Inf. Technol., vol. 15, no. 24, pp. 5159–5162, 2016.

[51] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "MIFS-ND: A mutual information-based feature selection method," Expert Syst. Appl., vol. 41, no. 14, pp. 6371–6385, 2014.

[52] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," J. Mach. Learn. Res., vol. 5, no. Oct, pp. 1205–1224, 2004.

[53] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," Neurocomputing, vol. 70, no. 1–3, pp. 489–501, 2006.

[54] H. A. Wibawa, I. Malik, and N. Bahtiar, "Evaluation of Kernel-Based Extreme Learning Machine Performance for Prediction of Chronic Kidney Disease," in 2018 2nd International Conference on Informatics and Computational Sciences (ICICoS), 2018, pp. 1–4.