Accepted Manuscript

Big Data Model Simulation on a Graph Database for Surveillance in Wireless Multimedia Sensor Networks

Cihan Küçükkeçeci, Adnan Yazıcı

PII: S2214-5796(17)30010-2
DOI: https://doi.org/10.1016/j.bdr.2017.09.003
Reference: BDR 76

To appear in: *Big Data Research*

Received date: 16 January 2017
Revised date: 14 June 2017
Accepted date: 14 September 2017

Please cite this article in press as: C. Küçükkeçeci, A. Yazıcı, Big Data Model Simulation on a Graph Database for Surveillance in Wireless Multimedia Sensor Networks, *Big Data Res.* (2017), https://doi.org/10.1016/j.bdr.2017.09.003

Big Data Model Simulation on a Graph Database for Surveillance in Wireless Multimedia Sensor Networks

Cihan Küçükkeçeci[a], Adnan Yazıcı[a]

[a]*Middle East Technical University, Department of Computer Engineering, Ankara, Turkey*

Abstract

Sensors are present in various forms all around the world such as mobile phones, surveillance cameras, smart televisions, intelligent refrigerators and blood pressure monitors. Usually, most of the sensors are a part of some other system with similar sensors that compose a network. One of such networks is composed of millions of sensors connect to the Internet which is called Internet of things (IoT). With the advances in wireless communication technologies, multimedia sensors and their networks are expected to be major components in IoT. Many studies have already been done on wireless multimedia sensor networks in diverse domains like fire detection, city surveillance, early warning systems, etc. All those applications position sensor nodes and collect their data for a long time period with real-time data flow, which is considered as big data. Big data may be structured or unstructured and needs to be stored for further processing and analyzing. Analyzing multimedia big data is a challenging task requiring a high-level modeling to efficiently extract valuable information/knowledge from data. In this study, we propose a big database model based on graph database model for handling data generated by wireless multimedia sensor networks. We introduce a simulator to generate synthetic data and store and query big data using graph model as a big database. For this purpose, we evaluate the well-known graph-based NoSQL databases, Neo4j and OrientDB, and a relational database, MySQL. We have run a number of query experiments on our implemented simulator to show that which database system(s) for surveillance in wireless multimedia sensor networks is efficient and scalable.

Keywords: Internet of things (IoT), big graph databases, NoSQL databases, wireless multimedia sensor networks, simulator

1. Introduction

A wireless multimedia sensor network (WMSN) is a distributed wireless network that consists of a set of multimedia sensor nodes, which are connected to each other or connected to leading gateways. Nowadays, smart devices such as mobile phones, smart televisions, and smart watches are equipped with sensors and network connections. Hence, with the advances in wireless communication technologies, multimedia sensor networks are expected to be one of the major components in the Internet of things (IoT).

A typical application for a WMSN is a surveillance system or a monitoring system. Smart city surveillance cameras with 7/24 recording, or one million sensor nodes reporting meteorological data produce data in various formats as video, audio, and text [1]. All that huge structured or unstructured data is considered as big data, which is defined by a number of Vs; *Volume, Velocity, Variety, Veracity, and Value*. Min et al. [2] present a comprehensive survey of big data and they identify that "defining the structural model of big data" is a fundamental problem. Fusing and analyzing big data are challenging tasks and there are many research studies that are related to big

Email addresses:
cihan.kucukkececi@ceng.metu.edu.tr (Cihan Küçükkeçeci), yazici@ceng.metu.edu.tr (Adnan Yazıcı)

data from different points of view in recent years. As pointed out by many researchers, relational database management systems (RDBMS) are inadequate for efficiently handling big data; therefore NoSQL database systems are mostly utilized [3, 4, 5]. There are four main types of NoSQL databases, which are key-value store (e.g. Amazon's Simple DB), big table (e.g. Apache Cassandra), document store (e.g. MongoDB) and graph-based model (e.g. Neo4j) [6].

Graph databases consist of nodes and edges (relations between nodes) which store data as properties. Graph databases are very efficient and convenient to handle social networks, fraud detection, graph-based operations, real-time recommendations and hierarchical relations. While storage of the big data is an important task, processing the streaming data and taking action for mission-critical applications are crucial. Arkady et al. [7] state that analyzing and extracting the valuable data from dirty raw data is an important research topic. In order to process that kind of data, we need to identify the data flow.

In this paper, we propose to use a graph-based model for handling big data generated from surveillance applications in wireless multimedia sensor networks. For this reason, we propose a graph-based model as a generic model to be used for different surveillance applications. Big sensor data is stored in a graph database for the purpose of advanced analytics, such as data mining, prediction, and statistics. Our graph model represents both the data flow among the nodes and wireless multimedia sensor network topology. The applicability of our solution is illustrated with a prototype implementation including simulation of synthetic data. A case study in the military surveillance domain is simulated and several experiments are done to measure the efficiency of our solution. Simulation results show that our proposed multimedia wireless sensor network model is applicable in large-scale real-life application scenarios.

The contribution of our study is to store the multimedia sensor network data in a well-defined graph-based big database model. The big data stored in an open-source graph database can be used for analyzing, filtering, aggregating and correlating big data. A simulation infrastructure is implemented for simulating multimedia wireless sensor networks to run a number of complex experimental queries. Although there have been some related studies in literature about the surveillance systems in the big data context, to the best of our knowledge, there has not been any applicable graph-based big data model for WMSNs based on a graph database yet.

This article is organized as follows: next two sections provide background information and related work. Then we introduce our real WSN system to give technical details of our deployment. Section 5 is the proposed graph-based big model explanation and Section 6 presents the prototype implementation of our model. Simulation infrastructure of wireless multimedia sensor networks is given in Section 7. Section 8 illustrates a case study in the military surveillance domain and Section 9 presents the experimental results and evaluations. Finally, conclusions are drawn in Section 10.

2. Background

2.1. Internet of Things

The Internet of Things (IoT) is used by Kevin Ashton in 1999 [8] and it roughly means that the Things use the Internet instead of Humans. The sensors, RFIDs, and nanotechnology help this mission to be accomplished by taking away the need for human-entered data.

Pankesh et al.[9] propose a domain model for IoT to make a common understanding. To define the model, they reference to the real world applications and summarize under three headings; Intermittent Sensing, Regular Data Collection, and Sense-Compute-Actuate loops.

As parallel to Sense-Compute-Actuate cycle, Sensor-as-a-Service (Senaas) notion is defined by Sarfraz et al. [10]. They virtualized the sensors as services by an abstraction on technical details of sensors. They trigger services with an event in the sensor and compute it to reply an action. Their IoT virtualization framework is validated by a case study.

Atzori et al. [11] prepare a comprehensive survey about IoT. They identify the enabling technologies as sensing, identification and communication systems like RFID, WiFi, and sensors. In addition, middleware applications using Service Oriented Architecture (SOA) are important for data distribution. In the end, they list open issues such as privacy, addressing of things and non-standardized applications for the future.

IoT is fully connected to sensor technology and the researches about sensor networks directly or indirectly improve the IoT. Hong et al. [12] propose an approach to

IoT using IP-based wireless sensor networks. They realize that IoT probably has the same problems that Internet itself had in the past. So, they identify problems like IPv6 adaptation, mobility, web enablement, global time synchronization, and security. They also share evaluation results of an implementation of their proposed SNAIL (Sensor Networks for an All-IP World) platform.

2.2. Sensor Networks

A set of sensors called sensor nodes connected to each other or connected to leading gateways is simply called a sensor network (SN). If sensors have capability of collecting multimedia data and have a communication infrastructure among the sensors, it is called multimedia sensor network (MSN). If sensors are connected to each other using wireless technology then it is a wireless multimedia sensor network (WMSN).

Akyıldız et al. [13] discuss the state of the art of research on WMSNs as well as the challenges. The challenges related to WSN deployment configurations are summarized by Perera et al. [14] in their research. Another survey paper [15] enlists the challenging issues to design middleware systems for WSN. Some of the identified challenges are as follows: data fusion, resource management, scalability and network topology, security, Quality of Service and limited power.

Peng et al. [16] propose a wireless sensor network in which sink node is replaced by a cloud. They called sink point instead of gateway. Their simulation results show that cloud based architecture increases the WSN performance.

Arati et al. focus on the information retrieval from sensor networks and propose a hybrid protocol which is called APTEEN [17]. They make experiments by executing queries to show that proposed protocol performs better.

From the database point of view, Ramesh et al. [18] define a database layer on top of sensor network so that a database query is mapped to traversing sensor nodes in the WSN.

2.3. Big Data

The buzz word of the recent years, Big Data, defined by a number of Vs; Volume, Velocity, Variety, Value and Veracity as shown in Figure 1.

Volume is the quantity of stored data. Velocity is the speed of data generation or processing. Variety is the type and structure of the data. Value is the importance of information that data provides. Veracity is the variation in quality of data and inconsistency and uncertainty of data.

The survey paper [2] points the relation between IoT and big data. For example, jet aircraft engines produce one terabyte of data per flight using various sensors. Think about a huge number of flights in a day all around the world and then you can have really big data. HP prepared a business-value white paper related to big data. According to the paper, one trillion sensors, roughly 150 sensors for every person will be existed by 2030. The generated data will be mostly unstructured data and the value of it depends on how the information is extracted from the data. As the number of sensors increases,much more storage and processing will be required. And all of these create some new challenging issues.

Many papers [6, 4], state that the relational database management systems (RDBMS) are inadequate for big data and NoSQL database systems are a solution at least for time being.

2.4. NoSQL Graph Databases

Moniruzzaman et al. [3] evaluate NoSQL databases in the aspect of big data analytics. In their survey, they enlist different types of NoSQL databases according to characteristics (features and benefits of NoSQL databases), classification (key-value, document, column-based and graph); and evaluation with a matrix on the basis of few attributes like design, integrity, indexing, distribution, and system.

There are four types of NoSQL databases which are key-value store (e.g. Amazon's Simple DB), big table databases (e.g. Apache Cassandra), document-oriented (e.g. MongoDB) and graph databases (e.g. Neo4j). Graph databases [19] consist of nodes and edges (relations between nodes) which store data as properties. Most of graph databases provide the capability to label nodes and edges. NoSQL Graph databases provide many ways to query data;

- User interface via SQL-like query language (Cypher for Neo4j, SQL for OrientDB)
- User interface via Graph visualization to interact with the nodes and edges

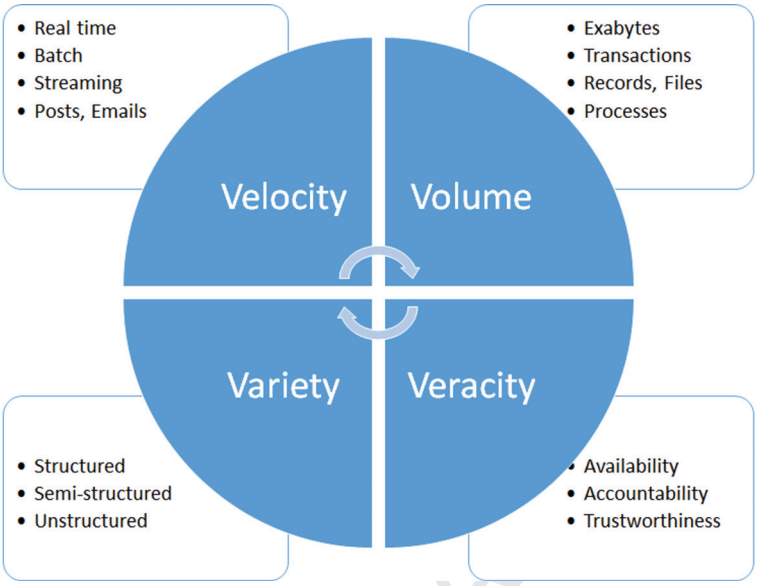

Figure 1: 5 Vs of Big Data

- Application program interface (API) to programmatically connect to database

Unfortunately, there is not any standardized way of querying, so that you have to write database specific queries every time. There is an open-source framework Apache TinkerPop to provide graph computing capabilities for graph databases. Gremlin is a part of the TinkerPop to traverse the graphs. And by the support of most of the graph databases, any gremlin query can be written once and works on every graph database.

Graph databases are generally preferred to handle social networks, fraud detection, graph-based operations, real-time recommendations and hierarchical relations.

3. Related Work

Over the years, various methods have been used for wireless sensor networks data representation and management ([20, 21]). Our approach is different basically by the aspects of big data, graph database storage, and our unique graph data model.

Yang et al. [22] present a service platform which is called Wiki-Health. They have designed platform in three layers; application, query and analysis, and data storage. In data storage layer, they used a NoSQL database as we do. But they use HBase which is a column-oriented key-value store, we use graph database which is better for relational analytics.

Christine et al. [23] discuss big data with spatial data received from wireless sensors using real life scenarios. One of the scenarios is related to smart cities which is similar to surveillance domain. They propose a scalable solution using Hadoop and HBase NoSQL database to prototype a platform for storage and processing wireless data. We also design and implement a simulation prototype from storage layer to analytics layer and more importantly, we propose a grah based data model on top of that architecture.

Renzo et al. [24] present a survey paper on graph database models. They compare graph database models with the other database models, i.e. a relational model. In this paper, we also compare well-known graph database models with the relational database model. Furthmore we perform queries to benchmark the performance of databases.

Another survey paper is a written by Felemban [25] which is about border surveillance. His research en-

lists the literature for experimenting work done in border surveillance and intrusion detection using the technology of WSN. Our research differs from the existing works by employing a graph based approach for surveillance domain and focusing on the simulation of the big data.

PipeNet [26] is a multi-layered wireless sensor networks application focused on pipeline monitoring. System aims to detect the leaks and other anomalies in water pipelines. They have used various types of sensors like pressure, pH and ultrasonic sensors on top of Intel Moto platform. Our sensor nodes are built on Rasberry Pi platform and have seismic, acoustic, and PIR sensors but also a multimedia camera. A camera needs further analysis like image processing and feature extraction. Their multilayer architecture is similar to our prototype but in another domain with different sensors and different analytical approaches. They try to analyze the collected multi-modal data for detection of the leaks. But we try to identify objects and track their movement. On the other hand, we approach our sensor data as big data.

Suvendu Kumar et al. [27] propose an analytic architecture for big data to detect intruders using camera sensors. Our work differs by using additional scalar sensors like acoustic and seismic sensors but also proposed graph based data model.

4. Real WSN System

Our reference WMSN system is composed of wireless multimedia sensors and some scalar sensors. The system is designed as a multi-tier automated surveillance system. The first layer is the sensing layer with scalar sensors including acoustic, seismic, and PIR. The second layer is triggered by first layer. Multimedia sensors like camera and microphone are used capture video and audio. After applying the fusion, object type and location of the sensor is extracted to be provided to the next layer, which is called the sink layer. The sink layer provides the capability to do analytics on all collected and generated information in the network.

The overall architecture is given in Figure 2. Sensor nodes are connected to the gateway nodes via ZigBee (IEEE 802.15.4) interfaces. A special thread for serial messaging is developed to send the events from sensor nodes to the gateway. The gateway prepares XMPP messages using the gathered events coming from leading nodes via broadcast messages. Those XMPP messages are transferred to the sink node. The XMPP messages and multimedia data is transfered over IP based (IEEE 802.11) connection.

The hardware of our sensor node is based on a Raspberry Pi (RPi) 512-MB Model B board and includes the

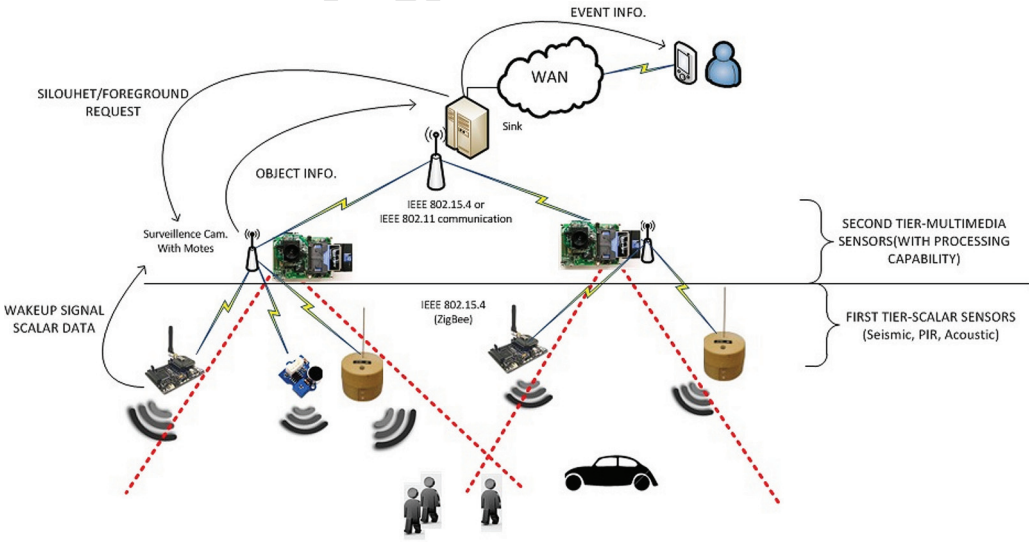

Figure 2: Real WMSN System Architecture

following hardware components:

- ARM1176 700MHz processor,
- Graphical processing unit (GPU),
- 512 MB SDRAM shared with GPU,
- SD card slot for on board storage,
- On board 10/100 Mb Ethernet port,
- 2x USB 2.0 ports,
- 1 CSI input connector for the camera module,
- Video and audio outputs,
- GPIO ports,
- 5V 700-mA microUSB power requirement.

The following list is the components installed on a node in our WSN system to fulfill its functions:

- Motion sensor (PIR),
- Acoustic sensor (AS),
- Vibration sensor (VS),
- Raspicam camera module,
- Xbee ZigBee (IEEE 802.15.4) adapter,
- 4400-mAh 5V 1A power bank,
- Microphone,
- Wi-Fi (IEEE 802.11) dongle,
- XMPP client software

Sensor to sensor communication, as well as gateway to sink communication, is completed via ZigBee interfaces. Nodes are equipped with low-bandwidth radio devices using IEEE 802.15.4 (ZigBee) standard and ZigBee provides a line of sight up to of 1500 m. at outdoor conditions and 250 Kbps at most.

In our real WSN system, sensor node and gateway roles are all predefined, there is not any dynamic gateway selection. Because different roles may need different kinds of hardware components.

5. The Graph-Based Big Data Model

Our graph-based big data model is built on the multimedia sensor networks topology. There is a sink node in the base station and there are a number of clusters connected to the sink. Each cluster consists of a gateway, which can be called the cluster head, and a set of sensor nodes.

The data flow occurs from the sensor nodes to gateways and from gateways to the other gateways (multi hop) and finally to the sink node. Each sensor node holds a set of data sensed by the sensors and camera of the node. Sensor nodes include embedded programs for handling correlation, transformation, and aggregation on the raw data, which is called first level fusion. The sensor fused data are reported to the leading gateway by all of its connected sensor nodes.

The gateway waits for all sensor nodes to report. When all reports are ready, the gateway applies an aggregation or filtering on the received data. The second-level fusion is done at this point and the output of the fusion is a summary of that cluster. The gateway fused data are forwarded to the sink node for a final decision.

Similar to the second-level fusion, the sink waits for all gateways' fused data. By applying some patterns to detect anomalies or other kinds of analysis are done at the third level fusion. The output of the last level fusion is an action like triggering an alarm or a notification message to another system.

Figure 3 shows the first level fusion graph model from raw data collection to fusion. The sensor node is responsible for fusion at this level. The input is the raw data and

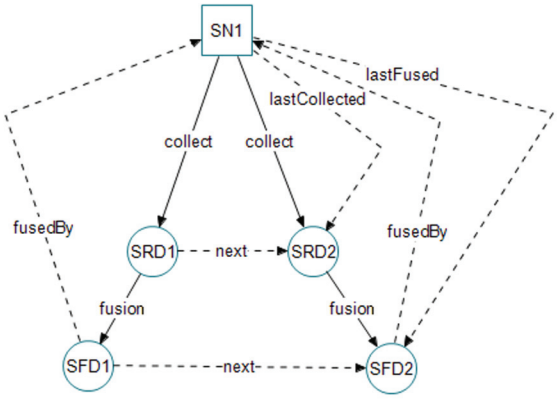

Figure 3: First Level Fusion model (SN:Sensor Node, SRD:Sensor Raw Data, SFD:Sensor Fused Data)

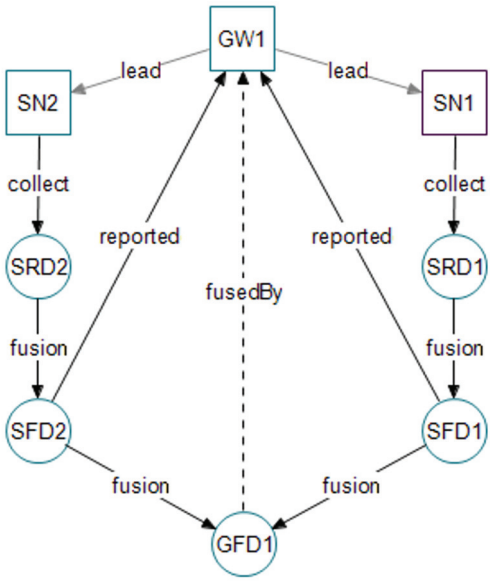

Figure 4: Second Level Fusion model (GW:Gateway, SN:Sensor Node, SRD:Sensor Raw Data, SFD: Sensor Fused Data, GFD:Gateway Fused Data)

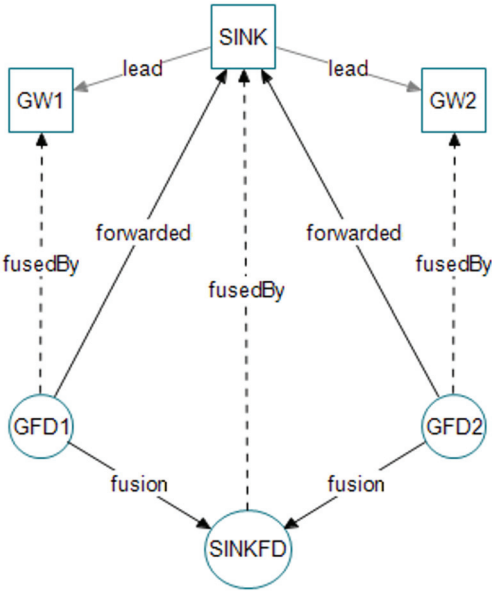

Figure 5: Third Level Fusion model (GW:Gateway, GFD:Gateway Fused Data, SINKFD:Sink Fused Data)

the output is sensor fused data. Last collected raw data and last fused fusion data are explicitly pointed for the purpose of the direct link.

Figure 4 shows the second level fusion graph model to identify the relations between sensor nodes and the gateway. The gateway is responsible for the fusion at this level. The input is all sensor fused data coming from sensor nodes and the output is gateway fused data which can be filtered, aggregated or transformed data.

Figure 5 shows the third level fusion graph model which is executed at the sink to aggregate all data fused from gateways. The input is the all fused data coming from gateways and the output is sink fused data which cause actions like triggering an alarm or notifying the operators.

6. Prototype Implementation for Proposed Model

We have already implemented the proposed graph model using OrientDB Graph Database and developed a simulator to generate synthetic data. In the following subsection, we describe why OrientDB graph database is chosen as default storage system. Other subsections show the implementation of the data model with the detailed design of generic infrastructure and simulation infrastructure.

6.1. Graph Database Selection

Our research includes applying the currently available databases to our graph-based big data model. Salem et al. [28] compare a set of databases like Cougar and TinyDB. Li-Yung Ho et al. [29] propose a distributed graph database based on an open-source graph database which is called Neo4j. The options are limited if you are looking for a graph database. Neo4j, Titan, and OrientDB are featured open-source graph databases.

Neo4j is a well-known graph database and used by many researchers. In addition, Neo4j is relatively easier to be used rapidly by developing some small pieces of code. Spring Framework support is really helpful to put things together very fast.

Titan is another open-source option for a graph database but its development is stopped and discontinued in early 2015. Therefore we did not prefer to utilize Titan.

Table 1: Compare OrientDB and Neo4j Community Editions

Feature	OrientDB	Neo4j
Graph Database	Yes	Yes
TinkerPop Standard Compliance	Yes	Yes
ACID Transaction	Yes	Yes
Unique Constraints	Yes	Yes
Fulltext Support	Yes	Yes
Spatial Support	Yes	Yes
Java Hooks	Yes	Yes
Record Level Security	Yes	No
User and Role Security	Yes	No
SQL	Yes	No
Dynamic Triggers	Yes	No
Custom Data Types	Yes	No
Additional Constraint Types	Yes	No
Indexes on Multiple Properties	Yes	No
Different Schema Modes	Yes	No
Multi-Master Replication	Yes	No
Sharding	Yes	No
Elastic Scalability	Yes	No
Server-Side Functions	Yes	No
Embeddable with No Restrictions	Yes	No
Sequences	Yes	No

OrientDB is another open-source graph database which is not as popular as Neo4j for now but has many advantages over it. Table 1 shows the comparison of OrientDB and Neo4j Community Editions which is provided by the official website of OrientDB. From all those compared features, "Multi-Master Replication", "SQL" and "Elastic Scalability with Zero Configuration" are the most important features for us to choose OrientDB.

6.2. Data Model Implementation

Nodes (vertices) and relations (edges) are defined in graph databases to store data. Compared to the traditional RDBMS approach, every row in a table is replaced with a node and its properties. There are edges to represent cross-table references.

At the first step, we define the node and edge types. Node types are; Sink, Gateway, SensorNode, SensorRawData, SnFusedData (*SensorFusedData*), GwFusedData (*GatewayFusedData*) and SinkFusedData. Edge types are; Lead, Collect, LastCollected, Next, Fusion, FusedBy, LastFusion, Reported and Forwarded. The edge types are defined for the usage between specific nodes. Table 2 lists the edge types in our graph database.

Each sensor node has the capability to hold temporarily a set of data which are sensed by sensors like *PIR*, *seismic*, *acoustic* and camera. That capability is provided by an in-memory database. For the fusion at the sensor node level, this in-memory database is used as a cache to analyze the changes in scalar sensors and provide some additional data to the first level fusion. The algorithm of the first level fusion is shown in Algorithm 1. The fusion result is stored in the *fusedData* including the *video*, *silhouette*, *foreground* and low level *features*.

Algorithm 1: Sample first level fusion algorithm executed on sensor nodes

```
1  Function firstLF(PIR, seismic, acoustic, threshold)
       Input  : Boolean PIR identifies if there is a
                movement or not, two integers seismic
                and acoustic values are scalar data
                sensed by the node, threshold is used to
                identify that there is an object with
                enough sound and vibration on sensor
                node
       Output: fusedData is the fusion data involving
                the multimedia data
2      initialize fusedData
3      if PIR = true and seismic ≥ THRESHOLD and
        acoustic ≥ threshold then
4          fusedData.video = startVideoRecording() ;
5          fusedData.frame =
             selectFrame(fusedData.video);
6          fusedData.frmFeatures =
             findLowLevelFeatures(fusedData.frame);
7          fusedData.foreground =
             selectForeground(fusedData.frame);
8          fusedData.fgndFeatures =
             findLowLevelFeatures(fusedData.foreground)
9          fusedData.silhouette =
             extractSilhouette(fusedData.foreground);
10     return fusedData;
```

Table 2: Node and Edge Types

Edge Type	From Node	To Node
Lead	Gateway	SensorNode
Lead	Gateway	Gateway
Lead	Sink	Gateway
Collect	SensorNode	SensorRawData
LastCollected	SensorNode	SensorRawData
LastFusion	SensorNode	SnFusedData
LastFusion	Gateway	GwFusedData
Next	SensorRawData	SensorRawData
Next	SnFusedData	SnFusedData
Fusion	SensorRawData	SnFusedData
Fusion	SnFusedData	GwFusedData
Fusion	GwFusedData	SinkFusedData
FusedBy	SnFusedData	SensorNode
FusedBy	GwFusedData	Gateway
FusedBy	SinkFusedData	Sink
Reported	SnFusedData	Gateway
Forwarded	GwFusedData	Gateway
Forwarded	GwFusedData	Sink

Sensor nodes apply the first level fusion and reports the output *fusedData* to leading gateway. The gateway applies the second level fusion as in Algorithm 2. The purpose of fusion at the gateway is a refinement of the sensor fused data before sending to the sink node. As a sample refinement algorithm, removing the duplications and normalizing the data according to scalar data can be written.

The output of the second level fusion is reported to the sink node. As the final decision maker, the sink correlates the concepts forwarded by gateways and decides that if an *action* is necessary or not. If an action is required, notification of the operator is triggered as given in Algorithm 3. All those three fusion algorithms are sample algorithms to provide the proof of concept execution of all phases of the simulated environment.

All those three fusion algorithms are sample algorithms to provide the proof of concept execution of all phases of the simulated environment.

6.3. Generic Infrastructure

We have developed the whole system by using Java 1.8 as maven projects to use Apache Maven as the de-

Algorithm 2: Sample second level fusion algorithm executed on gateways

```
1  Function secondLF(fusedDataList, threshold)
       Input  : The list of fusedData reported by
                leading sensor nodes, threshold is used
                to identify the difference between two
                scalar value
       Output: Filtered list of fusedData by
                duplication removal
2      initialize an empty array filteredDataList
3      for i = 0 to fusedDataList.size do
4          current = fusedDataList[i];
5          previous = previous element of current;
6          diff = current.acoustic - previous.acoustic;
7          diffRate = current.acoustic *
            threshold/100;
8          if diff < diffRate then
9              mark current as duplicate and drop
10         else
11             add current to filteredDataList
12     return filteredDataList
```

Algorithm 3: Sample third level fusion algorithm executed on the sink

```
1  Function thirdLF(filteredDataList, threshold)
       Input  : The list of fusedData reported by
                reporting gateways, threshold is used to
                identify the difference between two
                scalar value
       Output: Actions generated by reported
                fusedData list
2      initialize an empty array actionList
3      for i = 0 to filteredDataList.size do
4          current = fusedDataList[i];
5          previous = previous element of current;
6          if current.acoustic > threshold and
            previous.acoustic > threshold then
7              action = createAction(current);
8              add action to actionList;
9              notify operator using action;
10     return actionList
```

Table 3: Defined Message Queues

Queue Name	Process	Queue Element
Scalar Data	Collected Raw Data	SensorRawData
Fused	Level 1 and 2 Fusion	SnFusedData
Forward	Forwarding to Sink	GwFusedData
Action	Level 3 Fusion	SinkFusedData

pendency management framework. To develop the application of our research project beyond OrientDB, we have designed a generic infrastructure which can be easily adopted to other database systems. To achieve that, we have developed business logic without any dependency to OrientDB.

There are two managers called DataManager and NetworkManager. DataManager defines the necessary interfaces to populate data for the underlying database. NetworkManager has the business logic to construct network topology and defines the necessary interfaces to create network entities for the underlying database.

As there is a data flow between sensor nodes and gateways and sink, in order to cope with the bottleneck of high throughput of streaming data, we position Apache ActiveMQ message queues between each process. We define message queues as seen on Table 3 below. Fusion logic is developed on top of messaging queues and there are 3 business logic handlers; Level1and2Fusion, GatewayForward and Level3Fusion.

7. Simulation

To develop and experiment the graph data model, we have developed a simulator which is mainly focused on data simulation but also supports network topology simulation.

7.1. Network Topology Simulation

For our network topology, we assume that nodes are distributed with a grid layout to the simulation area which is assumed to be square. Figure 6 shows a visualization of 16 sensor nodes in each cluster with a gateway in the middle and 9 gateways in total with Sink in the center. Between two sensor nodes, the distance is 10 units. Each cluster has 4 sensor nodes at one side and there are 3 clusters in an edge of the total area. So, we have 120x120 sized grid layout for our simulation.

7.2. Data Flow Simulation

Data flow simulation is started with Raw Data Generator which produces synthetic data with position, PIR, seismic and acoustic information. These data are sensed by sensor nodes which are close to the position. A sensor raw data object is created for each sensor node with the calculated sensor data according to the distance to the position of the raw data created by Raw Data Generator. Then the generic infrastructure explained in the previous section is used to simulate all processes of the data flow.

7.3. Simulator

Currently, we adopt our simulation infrastructure on OrientDB, Neo4j which is the main rival of OrientDB and MySQL for the relational to graph database comparison. Every simulation is replicated to all three databases simultaneously so that the same test environment is provided.

Figure 8 is the screenshot of our data simulator to generate the network topology in Figure 6. “(Re)Create Database” button cleans the database and generates sink, gateways and sensor nodes according to the given parameter related to node count and cluster count.

“Start simulation” button starts simulation by sending an entity from one of the edges of the area. Then, the

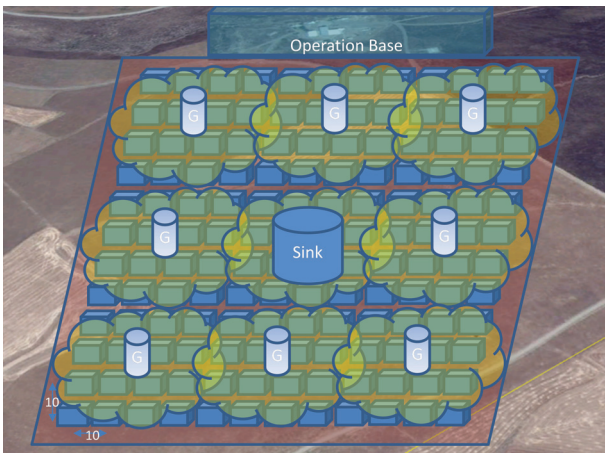

Figure 6: Simulated Network Topology

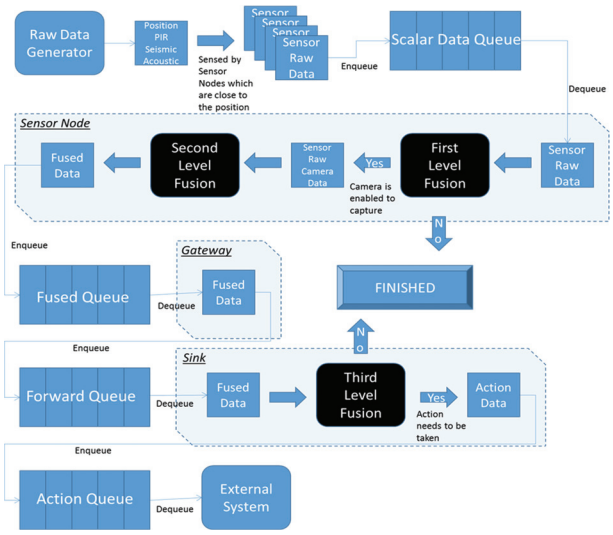

Figure 7: Data Flow Simulation

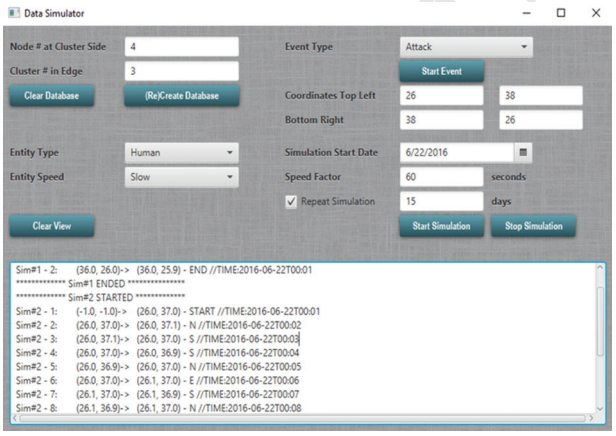

Figure 8: Data Simulator

entity moves randomly according to its speed and simulation ends when the entity moves out of the area. Possible moves are going to north, south, east, or west and don't move.

It is possible to run parallel simulations by clicking the button at any time. And if "Repeat Simulation" checkbox is checked, a new simulation is automatically started when current simulation ends.

There are several Entity types to simulate. Each entity has its own speed, acoustic and seismic values. Table 4 show each type and its simulation values. "Entity Speed" selection combobox decreases or increases the speed value of the simulating entity. Acoustic and seismic values are defined by the selection of "Entity Type" combobox and according to entity's distance from the sensor node, those values are recalculated to degrade its effect on the sensor node.

Assume that simulation put an "Animal" entity at (37,120) position. The entity is between the sensor nodes at (30,120) and (40,120). Table 5 and Table 6 show the sensed values between those two nodes for Animal Type.

Another important capability of the simulation tool is Event generation. We can identify event types and generate it at any time while the simulation is running. At first stage, we have 2 types of events.

- Attack: As seen in Figure 6, operational base is located at north. If something comes from south and directly moves toward the base, this movement is an Attack event for us.

- Smuggling: If a group of human is moving together with a group of animal and they are coming from west and going in the direction of south-east, this movement is smuggling event for us.

When "Start Event" button is pressed, selected event is

Table 4: Simulated Entity Types

Entity Type	Speed	Acoustic	Seismic
Human	1	20	10
Animal	2	40	20
Vehicle	4	70	80
GroupOfHuman	1	60	30
GroupOfAnimal	2	80	60

Table 5: Sensed Values at Node(30,10)

ANIMAL	30,10	31,10	32,10	33,10	34,10	35,10	36,10	37,10	38,10	39,10	40,10
Acoustic	20	18	16	14	12	10	8	6	4	2	0
Seismic	40	36	32	28	24	20	16	12	8	4	0

Table 6: Sensed Values at Node(40,10)

ANIMAL	30,10	31,10	32,10	33,10	34,10	35,10	36,10	37,10	38,10	39,10	40,10
Acoustic	0	2	4	6	8	10	12	14	16	18	20
Seismic	0	4	8	12	16	20	24	28	32	36	40

started to be simulated. Additional event types can be added for further analysis.

8. A Case Study: Surveillance Application

Surveillance systems need robust and scalable infrastructure. To achieve that, all data flow and data itself are needed to be analyzed and modelled.

A set of sensors and video cameras are needed to monitor the whole city. Assume that, we cluster the sensors and cameras according to the districts and each cluster forwards sensed data to the HQ (Head Quarter) which is the operation center. At the HQ, an alarm is triggered, or a notification is sent to the officers to early detection of violence.

Sensor types can be seismic, acoustic and PIR (Passive Infrared) which are types of scalar sensors. In addition to them, video cameras or thermal cameras can be added to critical locations. As the default, cameras are switched off. According to the sensed information from scalar sensors, the predefined conditions can be extracted using rule-based approaches. The motion or environmental change may be detected and interpreted to activate the camera by providing a rough prediction of the moving object.

We categorize the moving object as; Animal, Human, and Vehicle. The data collected from scalar sensors are analyzed to guess the category of the objects according to predefined thresholds (Table 7).

After activation of the camera by analyzing scalar sensor data, video and audio streams from the camera are started to be processed. That processing in the sensor node is called the first level fusion. After fusion, we have a concept of the moving object and maybe even a silhouette. Fusion output is reported to the gateway which is the leading node of that district.

A gateway is connected to a set of sensors and cameras. The described operations are done for all leaded sensors so that the gateway receives many concepts and silhouettes. By applying some algorithms like filtering the duplicate concepts or aggregating them to normalize the received data, the gateway accomplishes the second level fusion. After fusion, we have more accurate concepts and silhouettes provided by many sensors. Fusion output is forwarded to the HQ (Headquarter) for a final decision.

As there are many districts in a city, there are many gateways which are far away from the HQ and not directly connected to it. In that case, data is forwarded over other gateways. At the HQ, the received data from all districts are analyzed, aggregated or filtered for the purpose of detecting anomalies or finding some patterns, which is called the third level fusion. At the end, the fusion comes to the conclusion and an alarm is triggered to the officers or the external system of the armed forces is notified.

Table 7: Sample Thresholds To Identify Objects

Object Type	PIR	Seismic (Hz)	Acoustic (dB)
Animal	True	5 - 20	5 - 30
Human	True	21 - 55	31 - 50
Vehicle	True	>35	>50

9. Experimental Work

We setup a test environment to make some experiments on our graph model. Test environment specifications are;

- Intel i7-4710HQ Quad Core CPU
- 16 GB DDR3 RAM
- 240 GB SSD Storage
- 4 GB NVIDIA 860GTX GPU

Test environment has three database systems which are;

- OrientDB v2.2.5 (Graph database)
- Neo4j v2.3.2 (Graph database)
- MySQL v5.7.1 (Relational database)

We have simulated a multimedia wireless sensor network with synthetic raw data. Sensor nodes are placed in a square shaped area. Gateways are located in the center of each group of sensor nodes. The sink node is placed in the center of the whole area as shown in Figure 6.

There are 25 million sensor nodes which are leaded by 2,500 gateways. There is sink node to collect all data in our simulation environment. Each gateway leads 10,000 sensor nodes.

9.1. Comparison to Relational Data Model

This experiment is done on all three databases installed in our test environment. We have run many queries on both our graph data model and relational data model. Figure 9 shows the relational database representative of our graph data model on MySQL.

Below queries are randomly selected sample queries and Table IV shows the test results of the experiment.

9.1.1. Concepts Based Query

This query finds the specific type of objects with the highest probability of its detection time and location.

OrientDB Query:

```
SELECT concept, weight, fusionDate, out("fusedby").
    indexX, out("fusedby").indexY FROM fuseddata
    WHERE weight>0.90 AND concept = "Vehicle" ORDER
    BY fusionDate
```

Neo4j Query:

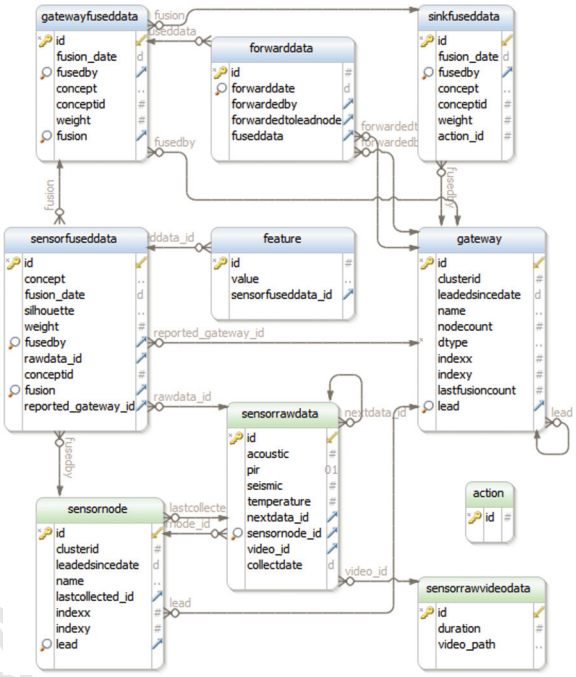

Figure 9: Relational Database Schema

```
MATCH (b:fuseddata)-[:fusedby]->(sd:sensornode) WHERE
    b.concept = "Human" AND b.weight>0.90 RETURN b.
    concept, b.weight, b.fusionDate, sd.indexX, sd.
    indexY
```

Relational SQL Query:

```
SELECT b.concept, b.weight, a.fusion_Date, sd.indexx,
    sd.indexy FROM sinkfuseddata a, gatewayfuseddata
    g, sensorfuseddata b, sensornode sd WHERE a.
    concept = 'Vehicle' AND a.weight>0.90 AND a.id =
    g.fusion AND g.id = b.fusion AND b.fusedby = sd.
    id ORDER BY a.fusion_Date ASC
```

9.1.2. Video Based Query

This query finds the possible explosions by identifying continued high volume around the surveillance area with their recorded video paths and video duration. The value bigger than 15 is assumed to be a high volume sound.

OrientDB Query:

```
SELECT in("collect").name[0], in("collect").indexX[0],
    in("collect").indexY[0], acoustic, out("video").
    videoPath[0], out("video").videoDurationSec[0]
    FROM sensorrawdata WHERE acoustic>15 AND out("
    next").acoustic[0]>15 AND out("next").out("next")
    .acoustic[0]>15 ORDER BY name
```

Neo4j Query:

```
MATCH (sn:sensornode)-[:collect]->(sa:sensorrawdata)
    -[:next]->(sb:sensorrawdata)-[:next]->(sc:
    sensorrawdata)-[:video]->(sv:sensorrawvideodata)
    WHERE sa.acoustic>15 AND sb.acoustic>15 AND sc.
    acoustic>15 RETURN sn.name, sa.acoustic, sn.
    indexX, sn.indexY, sv.videoPath, sv.
    videoDurationSec ORDER BY sn.name
```

Relational SQL Query:

```
SELECT s.name, s.indexx, s.indexy, ra.collectDate, ra.
    acoustic, v.video_path, v.duration FROM
    sensornode s, sensorrawdata ra, sensorrawdata rb,
     sensorrawdata rc, sensorrawvideodata v WHERE ra.
    acoustic>15 AND rb.acoustic>15 AND rc.acoustic>15
     AND s.id = ra.sensornode_id and ra.id = rb.
    next_id and rb.id = rc.next_id and rc.video_id =
    v.id ORDER BY s.name ASC
```

9.1.3. Recursive Query

This query finds the detected "Human" typed objects with high accuracy and calculates the distance of the sensor node to the sink node.

OrientDB Query:

```
SELECT $nodeId, out('fusedby').name[0], fusionDate,
    $deep.count FROM fuseddata LET $nodeId = out('
    fusedby').@rid, $deep = SELECT COUNT(*) FROM (
    TRAVERSE in('lead') FROM $nodeId) WHERE concept =
     'Human' AND weight > 0.9
```

Neo4j Query:

```
MATCH p=(a:sink)-[:lead*]->(b:gateway) WITH b.name as
    gname, length(p) AS depth MATCH (sfd:fuseddata)
    -[:fusedby]->(sn:sensornode)<-[:lead]-(g:gateway)
     WHERE sfd.concept = "Human" AND sfd.weight>0.9
    AND g.name = gname RETURN gname, sn.name, sfd.
    fusionDate, depth
```

Relational SQL Query:

```
WITH RECURSIVE search_graph(id, name, lead, depth) AS
    SELECT g.id, g.name, g.lead, 0 FROM gateway g
    WHERE g.lead is null UNION ALL
SELECT g.id, g.name, g.lead, 0 FROM gateway g WHERE g.
    lead is null UNION ALL
SELECT g.id, g.name, g.lead, sg.depth + 1 FROM gateway
     g, search_graph sg WHERE g.lead = sg.id
SELECT s.name, s2.name, s2.indexX, s2.indexY, s.depth
    FROM search_graph s inner join
select distinct sn.id, sn.name, sn.indexX, sn.indexY,
    sn.lead FROM sinkfuseddata sfd, gatewayfuseddata
    gfd, sensorfuseddata srfd, sensornode sn WHERE
    srfd.fusion = gfd.id AND gfd.fusion = sfd.id AND
    srfd.fusedby = sn.id AND
sfd.concept = 'Human' AND sfd.weight>0.9
s2 ON s2.lead = s.id
```

Table 8 shows the performance results of our example queries. For the first query is a simple range query which is focused on the basic query performance. OrientDB performs better than Neo4j because of the under-hood architecture of OrientDB which is a multi-model graph database. And graph model performs better than the relational model since there is no join operation as promised by the NoSQL databases.

Table 8: Test Results (Numbers are in milliseconds)

Query	OrientDB	Neo4j	MySQL
Concept Based	209	618	938
Video Based	355	145	422
Recursive	4,293	79,812	36,469

For the second query, Neo4j performs better than OrientDB and the graph model is again better than MySQL. This time again graph databases beat relational databases because of their join-free query capability. To understand why Neo4j is faster than OrientDB, we have to dive into queries. The query is neighbors and neighbors of neighbors query, which is a typical graph matching problem considering paths of length 1 or 2. In PostgreSQL we use a relational table with id from / id to backed by an index. Neo4j performs better because of its "index-free adjacency" for the edges.

The last query is to test the recursive SQL type of a query. The graph-based model is much faster than the relational model. Neo4j fails for this recursive query. There can be some optimizations possible while using the Cypher language (the query language of Neo4j) but we couldn't find them in the available Neo4j documentation.

9.2. Doubling Sensed Raw Data Size

An OrientDB graph database is selected for the execution of experiments. The previous experiments are applied on generated synthetic data of one month where each sensor node can sense data with 5 minutes of period. Now we increase the simulation duration from 1 month to 5 months step by step and diagnosed the query performance.

Figure 10 shows the chart of query times affected by the increased simulation duration. The query time is increased and it is better than linear which is fairly well compared to the doubled data size.

10. Conclusions

In this paper, we propose a graph-based model for complex multimedia and sensor big data. Our first aim is to represent the wireless sensor networks with multimedia data. Our implementation is composed of two main modules. The first module is the implementation of the

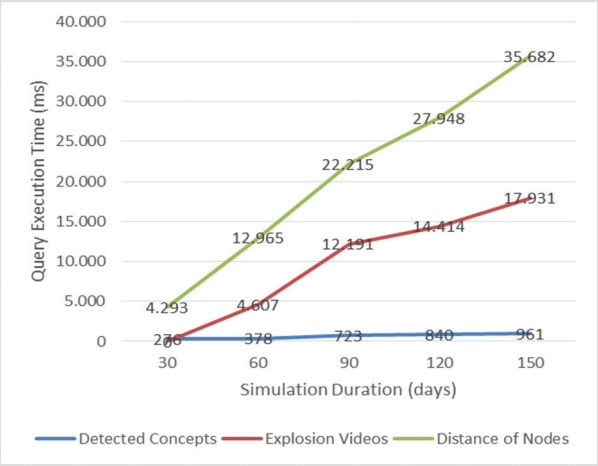

Figure 10: OrientDB Query Time/Simulation Duration Chart

proposed data model. The second module is the simulation which includes a simulator to produce synthetic big sensor multimedia data and the simulation infrastructure which represents the objects moving in our multimedia sensor networks.

We have focused on the surveillance systems to design our graph data model. The network topology and the data flow between each sensor nodes, gateways and sink are modeled so that all static and kinetic data is stored within the sensor network which must be assumed to be big data. The database to store big data is the NoSQL graph database. Because graph databases are good at representation of complex relations and scalable to store big multimedia sensor data.

We have tested our model on both graph databases and a relational database. Test results show that graph database model performs better that the relational database model. To decide which graph database is more convenient and efficient, we chose two well-known graph databases, OrientDB and Neo4j, for experiments. Neo4j is the market leading graph database, but it does not fit into our needs, which is to store complex multimedia big data. Because Neo4j supports high availability with the master-slave approach, which can scale vertically. But, in order to survive in the big data world, we need a master-master approach, which OrientDB has. In addition to that, our experiments show that OrientDB performs better than Neo4j.

We have simulated our WMSN prototype system with millions of data to test the proposed graph model simulating the system. We have tested the query performance with many complex scenarios. We have showed that our generated millions of synthetic data can be efficiently queried efficiently on our graph database.

As future research work, we plan to do graph analytics on our big graph model. Object tracking, topology optimization, and path extraction like analytics suit well for our surveillance domain.

Acknowledgements

This study is supported in part by a research Grant from TÜBİTAK with Grant No. 114R082. We thank to the researchers of CEng Multimedia Database Lab. for their very valuable comments.

References

[1] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Sensing as a service model for smart cities supported by internet of things, Transactions on Emerging Telecommunications Technologies 25 (1) (2014) 81–93.

[2] M. Chen, S. Mao, Y. Liu, Big data: a survey, Mobile Networks and Applications 19 (2) (2014) 171–209.

[3] A. Moniruzzaman, S. A. Hossain, Nosql database: New era of databases for big data analytics-classification, characteristics and comparison, arXiv preprint arXiv:1307.0191.

[4] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, D. Wilkins, A comparison of a graph database and a relational database: a data provenance perspective, in: Proceedings of the 48th annual Southeast regional conference, ACM, 2010, p. 42.

[5] Z. Xu, Y. Liu, L. Mei, C. Hu, L. Chen, Semantic based representing and organizing surveillance big data using video structural description technology, J. of Systems and Software 102 (2015) 217–225.

[6] J. Han, E. Haihong, G. Le, J. Du, Survey on nosql database, in: Pervasive computing and applications (ICPCA), 2011 6th international conference on, IEEE, 2011, pp. 363–366.

[7] A. Zaslavsky, C. Perera, D. Georgakopoulos, Sensing as a service and big data, arXiv preprint arXiv:1301.0159.

[8] K. Ashton, That 'internet of things' thing, RFiD Journal 22 (7) (2009) 97–114.

[9] P. Patel, A. Pathak, T. Teixeira, V. Issarny, Towards application development for the internet of things, in: Proceedings of the 8th Middleware Doctoral Symposium, ACM, 2011, p. 5.

[10] S. Alam, M. M. Chowdhury, J. Noll, Senaas: An event-driven sensor virtualization approach for internet of things cloud, in: Networked Embedded Systems for Enterprise Applications (NESEA), IEEE, 2010, pp. 1–6.

[11] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer Networks 54 (15) (2010) 2787 – 2805.

[12] S. Hong, D. Kim, M. Ha, S. Bae, S. J. Park, W. Jung, J.-E. Kim, Snail: an ip-based wireless sensor network approach to the internet of things, IEEE Wireless Communications 17 (6) (2010) 34–42.

[13] I. F. Akyildiz, T. Melodia, K. R. Chowdhury, A survey on wireless multimedia sensor networks, Computer networks 51 (4) (2007) 921–960.

[14] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, P. Christen, Sensor discovery and configuration framework for the internet of things paradigm, in: Internet of Things (WF-IoT), 2014 IEEE World Forum on, IEEE, 2014, pp. 94–99.

[15] X. Li, S. Moh, Middleware systems for wireless sensor networks: A comparative survey, Contemporary Engineering Sciences 7 (13) (2014) 649–660.

[16] P. Zhang, Z. Yan, H. Sun, A novel architecture based on cloud computing for wireless sensor network, in: Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering. Atlantis Press, 2013, pp. 472–475.

[17] A. Manjeshwar, D. P. Agrawal, Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks., in: Ipdps, Vol. 2, 2002, p. 48.

[18] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, S. Shenker, The sensor network as a database, Tech. rep., Citeseer (2002).

[19] I. Robinson, J. Webber, E. Eifrem, Graph Databases: New Opportunities for Connected Data, " O'Reilly Media, Inc.", 2015.

[20] O. Diallo, J. J. Rodrigues, M. Sene, Real-time data management on wireless sensor networks: a survey, Journal of Network and Computer Applications 35 (3) (2012) 1013–1021.

[21] B. Bostan-Korpeoglu, A. Yazici, I. Korpeoglu, R. George, A new approach for information processing inwireless sensor network, in: 22nd International Conference on Data Engineering Workshops (ICDEW'06), IEEE, 2006, pp. 34–34.

[22] Y. Li, C. Wu, L. Guo, C.-H. Lee, Y. Guo, Wikihealth: A big data platform for health, Cloud Computing Applications for Quality Health Care Delivery (2014) 59.

[23] C. Jardak, P. Mähönen, J. Riihijärvi, Spatial big data and wireless networks: experiences, applications, and research challenges, IEEE Network 28 (4) (2014) 26–31.

[24] R. Angles, C. Gutierrez, Survey of graph database models, ACM Comput. Surv. 40 (1) (2008) 1.

[25] E. Felemban, Advanced border intrusion detection and surveillance using wireless sensor network technology, International Journal of Communications, Network and System Sciences 6 (5) (2013) 251.

[26] I. Stoianov, L. Nachman, S. Madden, T. Tokmouline, Pipenet: A wireless sensor network for pipeline monitoring, in: 2007 6th International Symposium on Information Processing in Sensor Networks, IEEE, 2007, pp. 264–273.

[27] S. K. Mohapatra, P. K. Sahoo, S.-L. Wu, Big data analytic architecture for intruder detection in heterogeneous wireless sensor networks, Journal of Network and Computer Applications 66 (2016) 236–249.

[28] S. Hadim, N. Mohamed, Middleware: Middleware challenges and approaches for wireless sensor networks, IEEE Distributed Systems 7 (3) (2006) 1.

[29] L.-Y. Ho, J.-J. Wu, P. Liu, Distributed graph database for large-scale social computing, in: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, IEEE, 2012, pp. 455–462.