Design an Indexing Structure System Based on Apache Hadoop in Wireless Sensor Network

*Kongkea Keo, **Yeongjee Chung
Dept of Computer Engineering, Wonkwang University
e-mail: *k_kongkea@yahoo.com, **yjchung@wonkwang.ac.kr

Abstract

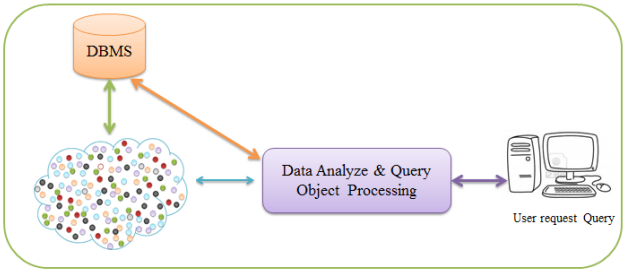
In this paper, we proposed an Indexing Structure System (ISS) based on Apache Hadoop in Wireless Sensor Network (WSN). Nowadays sensors data continuously keep growing that need to control. Data constantly update in order to provide the newest information to users. While data keep growing, data retrieving and storing are face some challenges. So by using the ISS, we can maximize processing quality and minimize data retrieving time. In order to design ISS, Indexing Types have to be defined depend on each sensor type. After identifying, each sensor goes through the Indexing Structure Processing (ISP) in order to be indexed. After ISP, indexed data are streaming and storing in Hadoop Distributed File System (HDFS) across a number of separate machines. Indexed data are split and run by MapReduce tasks. Data are sorted and grouped depend on sensor data object categories. Thus, while users send the requests, all the queries will be filter from sensor data object and managing the task by MapReduce processing framework.

1. Introduction

Every day, users create larger amounts of data compare to the past decade. All of these data came from different wireless sensor sources. For example sensor use for provide: weather information, daily life information inside the building, protect human life from fire and even forest fire sensor. In this research topic, I focused on six kind of sensors such as temperature, humidity, wind_speed, pressure, pollen, smoke as shown in Table1. Each of these sensors frequently updating themselves so how data is going to be stored, controlled and retrieved cause big challenges. A variety of systems architectures implemented for incremental data processing application including parallel and distributed relational database management system has been proposed (see on Figure 1). However, it still cannot meet what has been required. Therefore, ISS was introduced to increase the processing quality time based on Apache Hadoop. HDFS was introduced to store enormous data and work with MapReduce. MapReduce run indexed data in parallel and query user requested.

<Table 1> Sensor Type

Data Types	Unit
Temperature	°C
Humidity	%
Wind_speed	Km/h or m/s
Pressure	hPa or mbar
Pollen	Low/Medium/High
Smoke	°C and %



(Figure 1) Sensor data processing structure on DBMS.

2. Related Works

In this section, contributed the important reasons why different kind of sensors, and ISS were selected and studied based on Apache Hadoop. More complete and detail of these matters can be found in references information below.

2.1 Sensor Network Data Types Discovery

• Weather Information

Weather Information: Many sensor types have been proposed for example in paper [1] and [2] data have been collected such as: air temperature, pressure, wind speed, humidity and pressure in order to provide weather forecasting.

• Life Information

Life Information: In technique for modeling and learning statistical contextual information, the most general that can create highly efficiency life information inside the building are: temperature, humidity and pollen sensors [3] have been applied to our topic.

• Forest_Fire Information

Forest_Fire Information: In [4], there are temperature, humidity and smoke sensor involved in the research approach. Beside in [5], inputs four sensors: humidity, temperature, smoke and wind speed. Although many different kinds of sensors have been proposed, in this paper, we present temperature, humidity and wind_speed sensor that can use to protect the forest from fire.

• Fire Information Discovery

Fire Information Discovery: The studied on fire detection [6] and Other papers presented smoke sensor to control the fire. So in this paper, in order to provide fast and accurate fire alarm system, we use smoke sensor to detect temperature and humidity inside the building.

2.2 Indexing Structure Technique Discovery

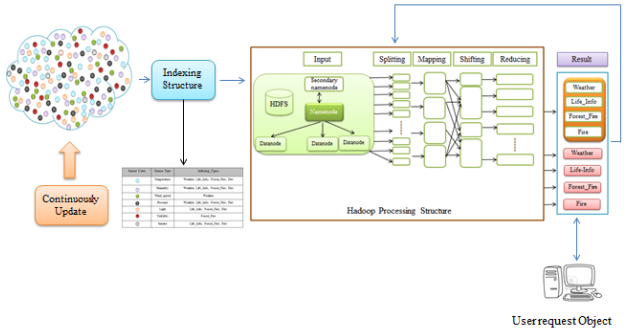
In the efficient updates for web-scale indexed over the cloud paper, they proposed an indexing system that can enable fast and frequent update on web-scale inverted indexes [7]. Thus in our paper, ISS was proposed and it has the similar ability like web-scale indexing system. For ISS can maximize processing quality and minimize data retrieving time.

2.3 Implementation Issue by Hadoop Processing

In the same paper [7] and [8], use MapReduce to create high performance, fully distributed index creation and update system. So in this paper, Apache Hadoop is the only solution that fit for large amount of data processing for it can store, retrieve and parallel processing large amount of indexed data.

3. Sensor Data Processing

WSN continuously keep sense and provide information to users. In order to take advantages of the available information, huge dataset have to be effectively indexed through ISS process. Each sensor keeps streaming their data and pours it into HDFS file system for data storing. HDFS has the responsibility keep data safe until retrieving time. I named data that already pass through indexing structure as indexed data. Thus these indexed data are split and done the map and reduce task in order to produce sensor data object, which means that indexed data are cover and put differently according to their ability (see Figure 2).



(Figure 2) Sensor data processing structure by using optimization indexing structure and Apache Hadoop.

4. Indexing Structure

4.1 Indexing Structure

As many sensors installed in different locations, each sensor is able to sense, collect, process and transfer the information to users. So in order to provide the accurate and specific information that requested by users, each sensor have to continuously update. After updating, all indexed data hold the sensing information, date, time and location. Therefore we can build the indexing structure table as shown below then information will be stored in data center. While each sensor keep generating, large amounts of dataset are mixed and lead to storing complicated. Especially, it take long time in every request. Thus indexing scheme provides innovative techniques that allow fast processing the new updating dataset. Furthermore, indexing structure also ensured that users always get what information they want and have fast access to newest information.

<Table 2> Indexing Structure

Indexed Data	Sensor1	Sensor2	.....	Date	Time	Location

4.2 Indexing types

According to related works above, we presented six different types of sensors such as: temperature, humidity, wind_speed, pressure, pollen and smoke that very common used nowadays. Each of these sensors has the potential creating and providing different task objects such as:

• Weather forecasting [1] and [2]: which provide very convenience information to users in checking daily temperature, humidity, wind_speed and pressure.

• Life_Info [3]: can provide the condition environment inside the building where we live and where we work. For example: temperature, humidity and pollen.

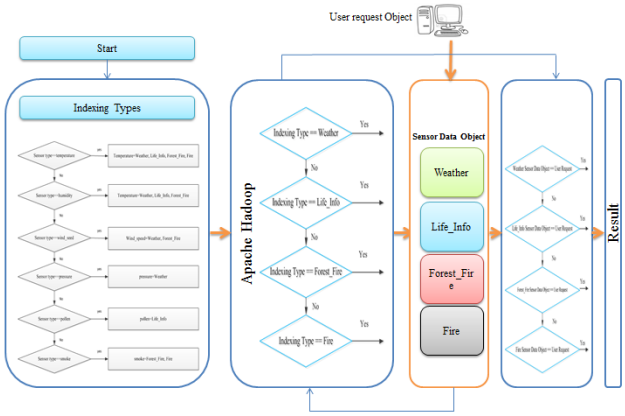
• Forest_Fire [4] and [5]: although in this two paper,

there were not exactly the same sensor types, all is still about how quick and well those sensors can perform and protect the forest from fire. So I decide to use temperature, humidity and wind_speed sensors for efficiency forest fire detection.

• Fire [6]: the past decade every people have feared and known how fire can destroy life. So smoke sensor was installed inside the building for detect temperature and humidity inside the house.

4.3 ISS Processing

Sensor senses and receives the information such as sensing information, date, time and location. Then each sensor passes through the ISS processing in order to get ready for the indexed. Each sensor get indexing depend on their own sensor type, we named this process indexing type. Indexing types have four categories such as Weather, Life_Info, Forest_Fire and Fire. For example: if sensor type is temperature sensor, then indexing type of temperature can be Weather, Life_Info, Forest_Fire, and Fire. Next is about Apache Hadoop processing. Sensor data will be store in HDFS and MapReduce will run the parallel processing and group the sensor by indexing type (Weather, Life_Info, Forest_Fire or Fire). For example: if indexing type is Weather, then there are temperatures, humidity, wind_speed and pressure sensor will involve in this category. After that we will get Weather, Life_Info, Forest_Fire and Fire sensor data object that include their own sensor types. Every time user send their query request, sensor data object use MapReduce for query processing in order to provide the final information to user. For instance: while user specifically sends the request asking about the weather information in one area, query will be done by using the weather sensor data object and run the MapReduce. MapReduce will provide the result to user depend on the condition requested by user. See the ISS processing on Figure 3.



(Figure 3) ISS processing.

5. Indexed data process in Apache Hadoop

While each indexed data constantly send, these large amounts of data need to be store and process.

5.1 Data Store in HDFS

Indexed data are store in HDFS in a redundant fashion across multiple machines. HDFS has one Namenode and many Datanodes.

• Namenode: is a master node. It can maintains indexed data tree, metadata and directories tree.

• Datanodes: are the worker nodes. They store, retrieve blocks when they are told to and report back to the namenode periodically with list of block they are storing.

5.2 MapReduce as Indexed Data Processing Model

MapReduce processes large volumes of data in parallel by dividing work into a set of independent tasks. MapReduce has one Jobtracker and many Tasktrackers. After retrieving data from HDFS, MapReduce will run data in different steps:

• Splitting: Indexed data is read from HDFS. Then each of input data is splitted into appropriate size.

• Mapping: generate indexed data key/value pairs that just splitted. All of them are run in parallel.

• Shuffling: transfer the map outputs to the reducers as input and sorted indexed data by keys.

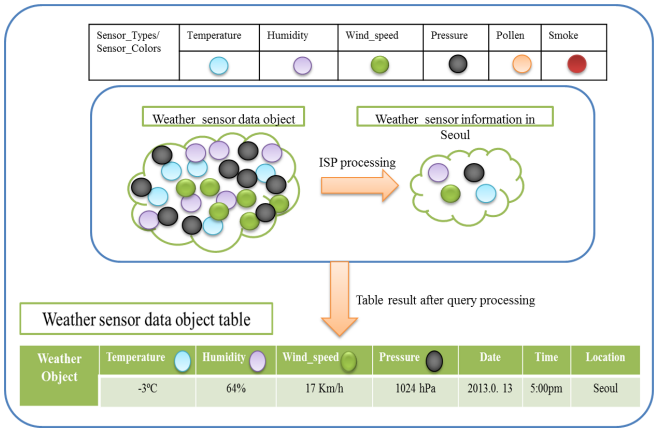
• Reducing: records with the same keys are handed by the same reduce tasks. Finally, result are written back to HDFS.

6. Sensor Network Data Object

6.1 Sensor Data Object

After the long process from indexing data and then indexed data are streamed and stored in HDFS. MapReduce process the running tasks and group indexed data into four categories such as: Weather, Life_Info, Forest_Fire and Fire sensor data object.

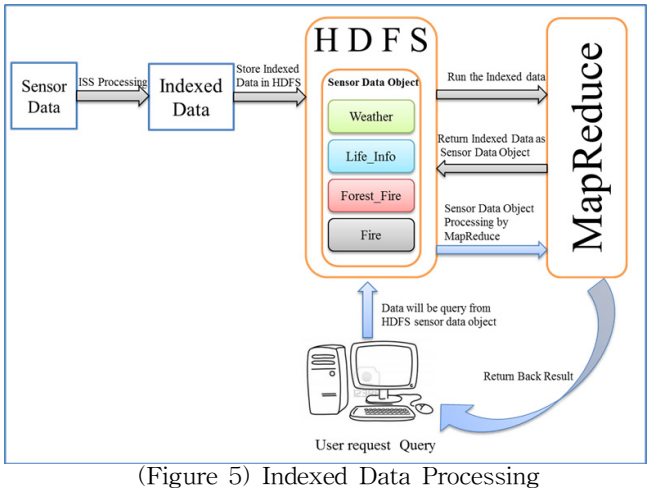
At this point, indexed data were grounded in a very proper simple manner. It is like we partition muddled sensor data and put it back in others repository. Hence, every time users send the requests, sensor data object will queries itself and do the MapReduce process. Base on indexing structure Table 2 above, Figure 4 below shows weather sensor data object information that includes sensor types and sensing information, date, time and location.



(Figure 4) Indexing Structure Object

6.2. Indexed Data Processing

Indexed data are named after sensor data pass through the ISS processing. Indexed data are processed by Apahce Hadoop. HDFS is the distributed file system and the place that indexed data can be stored. After finished storing, MapReduce has the responsibility to process indexed data in order to return sensor data objects. Sensor data objects are the group of indexed data and it use HDFS as data warehouse. Sensor data object will be generated if there is any user request. Result will be given by MapReduce parallel processing.



(Figure 5) Indexed Data Processing

7. Conclusion

In this paper, we proposed an Indexing Structure System (ISS) based on Apache Hadoop in Wireless Sensor Network (WSN). This ISS design can improve the data storing and processing quality. We use three different methods, giving Indexing Types to each sensor node, processing Indexing Structure and running indexed data on Apache Hadoop. By doing this, each sensor data are indexed. After indexing, indexed data store in HDFS and partition through MapReduce. After long run, indexed data were group into different sensor data objects. Even large amount of sensor node, it just takes very short time for every request. In this paper we only propose the design technique of ISS based on Apache Hadoop. In the future, a lot of works will be done such as: we will generate virtual sensors data and use MySQL as database storage. After that we will use DBInputFormat component which provided by Apache Hadoop to allow users interact sensors data with MySQL. MySQL provide JDBC driver that can allow Hadoop to connect to their database. Then we will perform Indexing Structure System. Data will be stored and analyzed on Apache Hadoop. As we have mentioned that MapReduce processing is use for big data so we will separately do the performance comparison between RDBMS and Apache Hadoop: for example by using Apache Hadoop, the amount of sensor data and time to run the request and answer it back to the user will strictly perform and compare to Database process. Also we will give the final conclusion that how many sensor nodes good for Apache Hadoop whereas how many sensor nodes good for RDBMS processing.

References

[1] Rogério B. et al: A Web Service-based Framework for Temporal/Spatial Environmental Data Access, 2012 12th International Conference on Computational Science and Its Applications.

[2] Yandong Wang, et al: A Framework of Spatial Sensor Web, Third International IEEE Conference.

[3] Eiman Elnahrawy, et al: Context-Aware Sensors, Springer-Verlag Berlin Heidelberg 2004.

[4] Liyang Yu, et al: Real-time Forest Fire Detection with Wireless Sensor Networks, 2005.

[5] Liyang Yu, et al: Real-time Forest Fire Detection with Wireless Sensor Networks, IEEE 2005.

[6] Mihai Marin-Perianu, et al: D-FLER: A Distributed Fuzzy Logic Engine for Rule-based Wireless Sensor Networks, 2008.

[7] Panagiotis Antonopoulos, et al: Efficient Update for Web-scale Indexes over the Cloud, IEEE 2012.

[8] XiLu Zhu, et al: Web Service Management Based on Hadoop, IEEE 2011.