International Journal of Smart Home
Vol.7, No.5 (2013), pp.39-48
http://dx.doi.org/10.14257/ijsh.2013.7.5.04

Design of Distributed Sensor Stream Data Input System

Seungwoo Jeon[1], Bonghee Hong[1], Joonho Kwon[2], Yoon-sik Kwak[3] and Seok-il Song[3]

[1] Dept. of Computer Engineering, Pusan National University
Jangjeon-dong Geumjeong-gu, Busan 609-735, Korea
{i2825t, bhhong}@pusan.ac.kr
[2] Institute of Logistics Information Technology, Pusan National University
Jangjeon-dong Geumjeong-gu, Busan 609-735, Korea
jhkwon@pusan.ac.kr
[3] Dept. of Computer Engineering, Korea National University of Transportation
50 Daehak-ro, Chungju-si, Chungbuk, 380-702, Korea
{yskwak, sisong}@ut.ac.kr

Abstract

Sensor networks are widely used in various environment monitoring systems. For example, in storehouse, it is used for monitoring the temperature, humidity, and carbon dioxide level inside the storehouse. Those sensing values are packaged in relatively small data, only around 100 bytes. However, we want to get those sensing values periodically every a certain cycle; every several seconds or minutes depending on the system requirement. And since in a storehouse sensor network there might be hundreds of storehouse and thousands of sensor nodes, resulting in millions of sensor data over only one day (depending on the cycle length), we are facing tremendous number of data access into the file system; using the naïve one-data one-file storing mechanism. In this paper, we are trying to design a data input system which collect and combine those individuals of sensor data into a big chunk of file and store it in HDFS, while using MapReduce framework to access and process those chunks inside HDFS.

Keywords: *Distributed sensor stream data, HDFS*

1. Introduction

There are so many sensor networks used in various environments. Usually, we use sensors to get the condition of the environments for monitoring purpose. Especially in logistics area, we need to monitor the products during the supply chain and maintain the quality of the products. For example, in the storehouse environment, we need to measure various environmental sensing value such as temperature, humidity, and carbon dioxide level. Furthermore, to get accurate monitoring data for effective decision support system information, we need to process those sensing values and keep guaranteeing the most recent result in real time.

A sensor network might consist of millions or billions of nodes. Each node will send the sensing value data to the hub every a certain period of time; each particular period of time is called a read cycle. Then the data will be sent to the server and processed further. The data will be processed right away as it comes either with or without storing process at the later stage. From computer networking point of view, this is called stream data processing.

The sensor data usually has a file size around 100 bytes. It is very small. However, the number of the sensor data from the whole sensor network for extended time period would be enormous, especially when the numbers of nodes are billions and the value of a single read cycle is short. When we store those data for historical data analysis, we will have a tremendously high data access into the file system. Therefore, we need a suitable data input system to handle the input data and store them into the file system.

We will utilize the Hadoop Distributed File System (HDFS) to store the input data and the MapReduce framework to access and process those data inside HDFS. In HDFS, we store the data in a fixed predefined-sized chunk of file, *e.g.*, 64 MB. We will collect and combine the small-sized individual stream data into a single large chunk then store it into HDFS. With this, the frequency of data access into the file system will be much more scalable. The remainder of this paper is organized as follows. Section 2 discusses previous works on HDFS. In Section 3, we define a problem in the target environment. In Section 4, we propose the system to solve the problem. A summary of the paper is presented in Section 5.

2. Related Work

2.1. Wireless Sensor Network

Wireless Sensor Networks (WSN) is a field that raises the interest of wide range of business domains [1-4]. It can be applied to various fields from the special application fields such as hazardous environmental monitoring, industrial machine measurement, and military-purpose measurement to the daily application fields such as fire monitoring and pollution monitoring. Considering the fact that Wireless Sensor Networks produce a large and diverse amount of data, business applications may have difficulties to select and process relevant information.

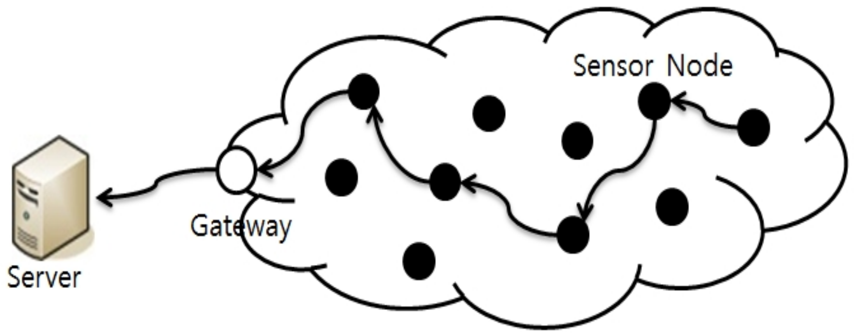


Figure 1. Wireless Sensor Network

Wireless Sensor Networks have a few to hundreds of nodes, where each node is connected to one or several sensors. WSN topology can be various such as it can be a simple network or an advanced multi-hop mesh network. No matter how complex the topology, WSN always consists of sensor nodes and base stations. Sensor nodes is small computer with limited computational power and limited memory, used for gather the information within its range. WSN base station is the name of one components with much more energy for computational and communication resources. They play role as a gateway between the end user and sensor nodes. WSN base station typically sends data from the WSN on to a server. Other special components in routing based networks are routers, designed to compute, calculate and distribute the routing tables.

2.2. Hadoop Distributed File System

Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware [5]. It is designed to be incredibly fault-tolerant and suitable for low-cost hardware. It works well for applications that have large data sets, and is able to provide high throughput access for the application data. HDFS is the open source implementation of GFS. It is also based on the large file, and therefore suffers performance efficiency in dealing with small files. The HDFS consists of a single master and multiple slave frameworks, when storing large amount of small files the master will accept requests for storage addresses and distribute storage block frequently.

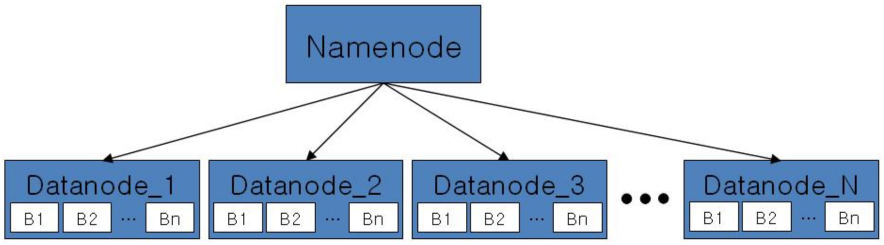


Figure 2. HDFS Architecture

Just like normal file system, HDFS has three kinds of file operations: write, read and delete. When a client need to data, client send a read request to the namenode for the list of datanodes that host replicas of the blocks of the file, then the client contacts a datanode. When client need to store data, it send a write request to the namenode. The namenode will generate a blockId and find the datanodes to save the data. Client will send the data to these datanodes according to data flow and notify the namenode to save the metadata if it successfully done.

2.3. Sensor Data Processing

With the fact that Wireless Sensor Networks produce a large amount of sensor data, business applications may have difficulties to select the relevant information to be processed. There are two methods that approach this issue: in-network processing and data processing within dedicated middleware (middleware processing) [1].

In-network data processing is the global process to reduce communication costs and avoid forwarding data that is of no use with the objective of reducing resource consumption. Because, in WSN, computation is typically consumes much less energy consuming than communication.

With data processing in a middleware, or middleware processing, any information from the WSN is forwarded and processed within a mediation layer, or middleware, between the WSN and business applications. Middleware processing does not consider energy saving within WSN, but rather provides multiple processing methods, matching business application needs for information.

3. Target Environment and Problem Definition

3.1. Target Environment

The target environment in this paper is storehouse management system for agricultural products using wireless sensor networks, which consists of numerous storehouses and a server. And storehouse is composed of lots of sensor nodes that are installed inside storehouse and measure conditions such as temperature, humidity, and carbon dioxide of adjacent space and a sensor hub which transmits sensing values from sensors to server. The server has a role for collecting and processing transmitted data from sensor hubs of each storehouse. Communication between sensor nodes and the sensor hub is to use short-range wireless technology, whereas communication between the server and the sensor hub is to use 3G technology.

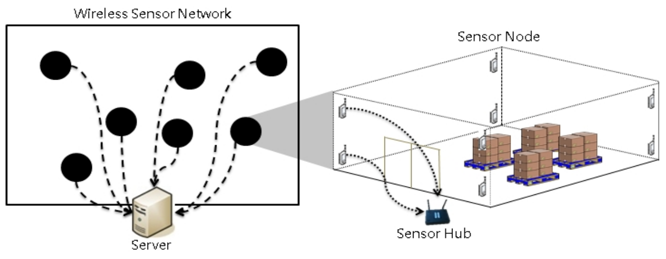


Figure 3. Target Environment

As mentioned above, Figure 3 shows how to be located in device component. The sensor nodes installed in inside storehouse transmit the data to the sensor hub using short-range wireless technology and the sensor hub also transmits the data collected on a regular cycle to the server using 3G technology.

3.2. Problem Definition

The existing distributed parallel processing systems are basically configured to fit the batch processing of the pre-collected big data. In other words, it is difficult for the current systems to process the stream data, where we are collecting the data continuously in real time. For example, as shown in the figure below, there are sensor data measured from numerous storehouses.

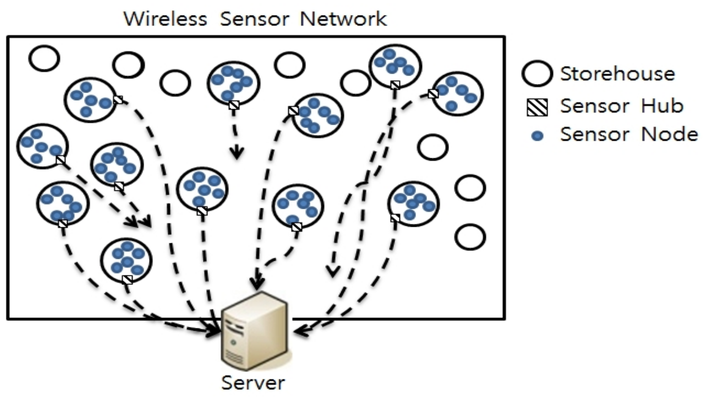


Figure 4. Concentrated-load Problem of Sensor Stream Data

Each sensor node transmits sensor data to the sensor hub and then the sensor hub transmits the received sensor data to the server. At this time, the number of sensor nodes in one storehouse can be dozens of nodes, at least. Moreover, there can be hundreds of storehouses, which lead to performance degradation of the existing system since it processes and stores individual sensor stream data one by one as they come. So, in order to process big data efficiently in the distributed system, we need to collect and store the data into several big chunks of file, which has a predefined maximum size, *e.g.*, 64 MB for every single chunk, and then store the data. Unfortunately, it is the opposite condition in the stream data processing system. Since the size of each individual sensor stream data is just around 100 bytes, it will make much more frequent access into the file system if those small chunks of file are stored individually; which obviously will lead to a significant performance problem.

4. Solution

4.1. System Architecture

To solve performance degradation due to frequent file system access, we propose distributed sensor stream data input system. This system should execute at least three tasks:

- Collect stream data within a given period and then transform a set of small stream data into one big chunk of file, since it is easy for MapReduce to find the data in HDFS.
- Manage to store stream data in order to prevent the missing data.
- Deliver the chunk to HDFS for storing stream data.

Figure 5 shows the system architecture about distributed sensor stream data input system and the system consists of Data Manager Component, Storage Component, Configuration Control Component, and Data Measurement Component.

- **Data Measurement Component**: The component measures sensor data generation rates per time slice and defines the amount of data transmitted.
- **Storage Component**: It consists of two temporary storages and stores the stream sensor data collected from Data Measurement Component. At this time, if one of temporary storages goes on to move transfer phase, the other collects the stream sensor data and controls the system for preventing the acquisition and transmission.

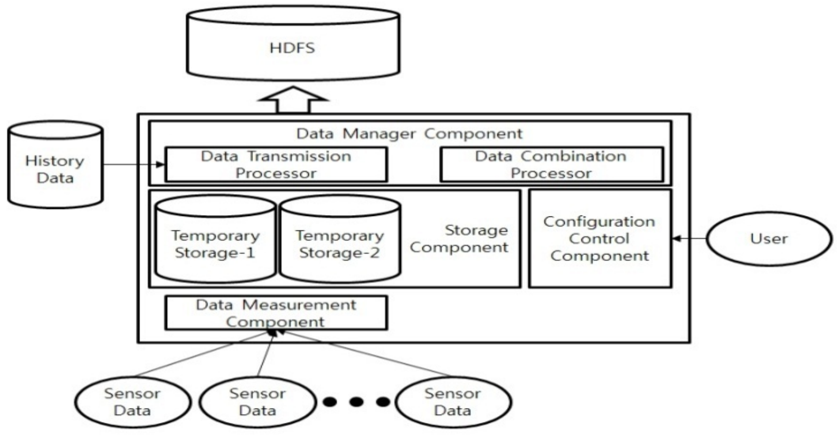


Figure 5. Distributed Sensor Stream Data Input System

Configuration Control Component: The component has a role of entering maximum waiting time to collect data from user.

- **Data Manager Component:** It is composed of two processors: Data Transmission Processor and Data Combination Processor. After collecting the stream sensor data as much as configuration values in Data Measurement Component, the component combines the data and transmits the combined data to HDFS.

Figure 6 shows the massive amount of continuously collected sensor stream data stored in HDFS.

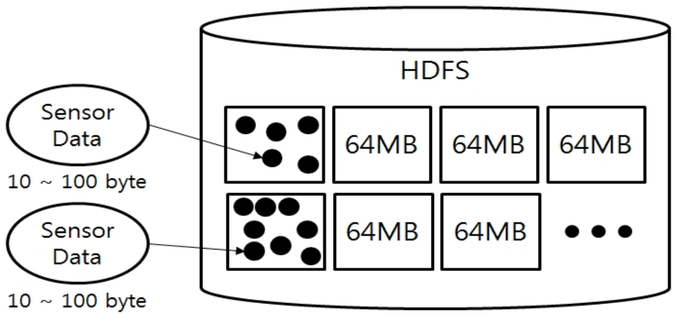


Figure 6. Sensor Steam Data stored in HDFS

The system figures out the volume of the collected sensor stream data in real time and accordingly, when the volume reaches the predefined chunk size or passed over a certain time limit, the collected sensor data are transformed into a chunk and transmitted to HDFS. Eventually, we can consider two ways as follows.

- **In case of huge amount of generated sensor data per time slice**: As mentioned above, since default chunk size is 64MB, if an enormous amount of the stream data per time slice is generated, then the system will transform the stream data into a default sized chunk (64MB) or over (>64MB). It preserves the performance of the system by preventing frequent file system access.
- **In case of tiny amount of generated sensor data per time slice**: For real time data processing, user firstly set the length of the time slice to collect the data. Right after passing the predefined time slice, the system will combine the collected data and transmits the combined data to HDFS.

4.2. Process of Input Stream Sensor Data

Figure 7 shows flowchart of processing sensor stream data inserted and consists of seven steps.

1. Enter configuration information from user: A user enters configuration information such as address of HDFS and maximum waiting time to collect using configuration control component.
2. Define data size to be transmitted: The system firstly measures sensor data generation rate per time slice through data measurement component and defines data size to be transmitted at a time.
3. Collect sensor data: The data to be transmitted from sensor nodes are entered to temporary storage.
4. Compare the amount of sensor data and configuration value: The data measurement component compares the amount of sensor data and configuration value. If the amount and the value are the same, then directly move to Combine step. If not, move to next step for comparing maximum waiting time.
5. Decide whether maximum waiting time is over or not: If maximum waiting time is over, then move to Combine step. If not, move to Collect sensor data step(third step) and repeat steps.

6. Combine the separated data: Before transmitting to HDFS, the separated data entered from fourth step or fifth step are combined using data combination processor in data manager component.
7. Transmit the combined data: The combined data are transmitted through data transmission processor in data manager component.

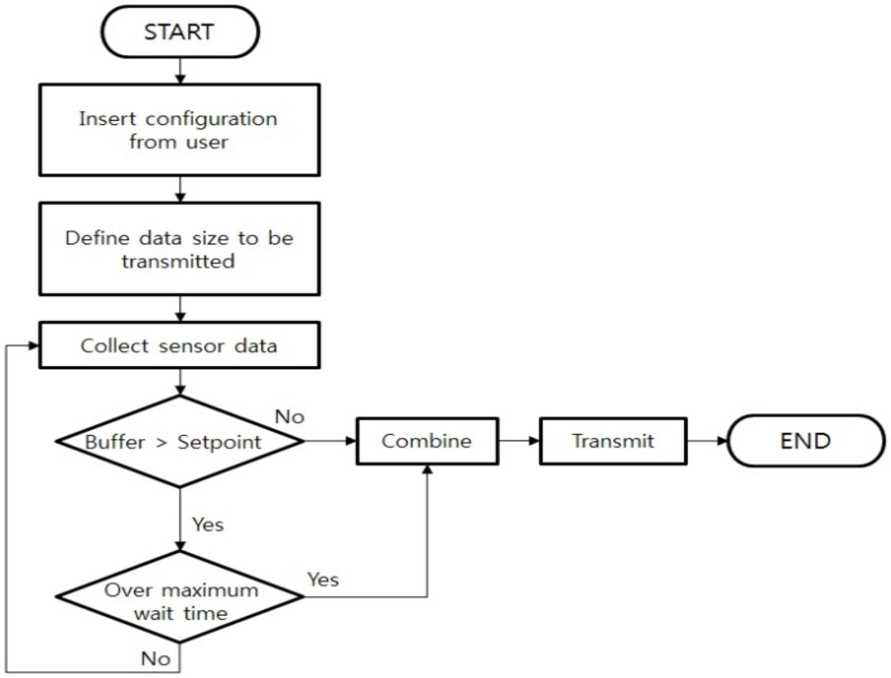


Figure 7. Flowchart of Sensor Stream Data Processing

For example, we consider a process of massive input data processing with proposed system in storehouse management system for agricultural products. If the number of storehouse is one thousand, each storehouse has one hundred sensor nodes and one hub, sensing time is three minutes, and the size of one packet to be transmitted is one hundreds byte, then about 200MB of real time data are generated per one hour. If the proposed system is not used in situation, then HDFS may have heavy load to process the data since HDFS should enter 200 million of the input data. Also, if user does not change the size of basic block units, HDFS can result in a tremendous loss of capacity. However, if uses our proposed system, there are no more frequent access since the system collects stream data to 64MB size or more.

5. Conclusion

In this paper, we are proposing a sensor data input system which collects and combines the individuals of stream data into predefined-size big chunks (of file). Its purpose is to reduce the frequency of data access into the file system thus maintaining the performance of the sensor data stream processing, making it much more scalable to handle an enormous amount of sensor stream data. The main steps are collecting, comparing, and combining. In the beginning, we are collecting the sensor data input stream into a buffer. After that, we are checking the collected data based on two characteristics. The first characteristic is the whole data size so far. If the data collected so far reaches the predefined chunk size, then combine and store the data right away. Otherwise, if the waiting time limit is over, then combine those collected data regardless of its smaller size (*e.g.*, data_size<64MB; predefined_chunk_size=64MB).

Acknowledgments

This research was supported by Bio-industry Technology Development Program, Ministry for Food, Agriculture, Forestry and Fisheries, Republic of Korea.

References

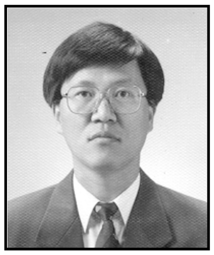
[1] L. Gomez and A. Laube, "Ontological Middleware for Dynamic Wireless Sensor Data Processing", Third International Conference on Sensor Technologies and Applications, **(2009)**, pp. 145-151.
[2] K. Martinez and P. Basford, "Robust wireless sensor network performance analysis", 2011 Sensors IEEE, **(2011)**.
[3] M. Guo, C. Lo, and K. Qian, "A Smart Routing Scheme for Wireless Sensor Networks", Wireless Information Technology and Systems (ICWITS), 2010 IEEE International Conference, **(2010)**.
[4] Y. Tian, B. Gao and L. Tong, "Design and Application of Wireless Sensor Networks for Ground Verification of Remote Sensing", Apperceiving Computing and Intelligence Analysis, 2008. ICACIA 2008. International Conference, **(2008)**.
[5] The Apache Software Foundation, HDFS 0.21Documentation, http://hadoop.apache.org /hdfs/docs/r0.21.0/.

Authors

Seungwoo Jeon received the MS degrees in Computer Engineering from Pusan National University, Busan, Korea, in 2011. He is Ph.D Candidate at Pusan National University, Korea. His current research interests include moving object databases, RFID system and RFID/RTLS/Sensor middleware.

Bonghee Hong received the MS and PhD degrees in Computer Engineering from Seoul National University, Seoul, Korea, in 1984 and 1988. In 1987, he joined the faculty of Computer Engineering of the Pusan National University (PNU). He is working as a Professor of Database in the Department of Computer Engineering at the PNU. He is a Director of Institute of Logistics Information Technology. He is also a steering committee member of DASFAA and was a co-chair of APWeb 2010. His research interests include theory of database systems, moving object databases, spatial databases, RFID system and RFID/RTLS/Sensor middleware.

Joonho Kwon received his PhD, MS and BS degrees in the School of Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea, in 2009, 2001 and 1999, respectively. He is an assistant professor of Institute of Logistics Information Technology at Pusan National University, Korea. His current research interests include XML filtering, XML indexing and query processing, Web services, RFID data management, multimedia databases and serious games.



Yoon-sik Kwak received his BS degree in the Electrical Engineering from the University of Cheongju in 1984, his M.S.E.E. degree from the University of Kyunghee in 1986 and his Ph.D. degree from the University of Kyunghee in 1994. He worked at Korea National University of Transprotation in the Department of Computer Engineering and rose to the level of Full Professor. His research interests are in the areas of signal processing, Internet communication, microcomputer system ,and applications of these methods to mobile system.

Seok-il Song received the BS, MS and PhD degrees in computer and communication engineering from Chungbuk National University in 1998, 2000 and 2003, respectively. Currently, he is an associate professor in the Dept. of Computer Engineering at Chungju National University, Chungbuk, Korea. His research interests include database management systems, concurrency control, high dimensional index structures, storage systems, moving object database, sensor network and XML database.