Redundant Data Removal Technique for Efficient Big Data Search Processing

Seungwoo Jeon¹, Bonghee Hong¹, Joonho Kwon², Yoon-sik Kwak³ and Seok-il Song³

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

Abstract
Ranch industry has grown bigger. In Australia, ranches have very large number of livestock commodities: cattle, lambs, and muttons. To manage such a very large scale commodities, they need to install sensor network with MapReduce of Hadoop; since the sensor network generates a huge amount of data. The ranch is divided into several patterned regions and a lot of hubs are installed in there for retrieving the sensor data. However, when the sensor moves to an overlapped area among some hubs, the sensor data are transmitted to all hubs covering the area. Obviously, these data is redundant. Therefore, we propose removal technique to delete the redundant data and to efficiently process the data on the map phase. In order to detect redundancy, the data will be compared using some parameters, and then the detected redundant data will be deleted according to some rules.

Keywords: Hadoop, MapReduce, Duplicate data, Removal technique

1. Introduction

Australia is one of the biggest meat producers, with 419 thousand tons of lamb; equals to 73.1 million head, and 2.1 million tons of beef; equals to 28.5 million cattle. In terms of export commodities, Australia is one of the world’s largest meat exporters as it exported around 116 thousand tons of mutton, more than 200 thousand tons of lamb and around 1.38 million tons of beef within the period of 2011-2012 [[1]]. No doubt, there are a lot of ranches in Australia that hold a large number of livestock on it.

Anna Creek Station, the biggest ranch in the world, covers 6 million acres of land in Australia. Assume that the grass is in the ideal condition where a cow needs 3-5 acres of it to live every day, then Anna Creek Station can hold more than 1 million cows or even more lambs and muttons. In such a very large scale of livestock, an effective monitoring system is required. Currently, there are several monitoring systems using RFID and wireless sensor technology, such as: Bella Ag [[2]], and eCow [[3]]. These systems works with typically several common components; RFID and sensor tag, hub, and monitoring application software. The tag is attached to the animal, the tag will send the sensing data to the reader/hub, and the reader/hub will send the data to the server computer, where finally the data will be processed and presented to the user. These monitoring systems detect the cow location, temperature, and several other values, according to the hardware specification, in real time.
The hubs are placed in the farm field with a certain approximation that all area in the field is covered. And since the coverage of a hub is the surrounding area within a certain radius, it forms a circle shaped coverage area. To be able to cover more complete area, some nearby hubs often overlap a certain coverage area between them. When the animal position is in this overlapped area, the sensor data will be transmitted to all hubs that cover it. This will produce redundant data and makes the stored data larger by several times. For example, if there are 1 million animals and each animal is in the overlapping area of 2 overlapping hubs for once, then half of the data are redundant.

Therefore, we propose a removal technique to eliminate this redundant data. We are detecting the redundant data by comparing the sensing values, hub ID, sensor ID and the timestamp parameters. We will also have a mapping of hub ID to region number. In this way, we can do comparison easier than comparing the hub ID string. We will only keep the entry which has the smallest region number and remove the other(s).

The remainder of this paper is organized as follows. Section 2 discusses previous works on the MapReduce of Hadoop. In Section 3, we define a problem in the target environment. In Section 4 and 5, we propose the technique to solve the problem and measure performance. A summary of the paper is presented in Section 6.

2. Related Work

A. MapReduce

Since it's firstly published in 2004, MapReduce was gaining more and more attention from all companies and researchers around the world. This craving for MapReduce was getting stronger after Doug Cutting created the open-source Hadoop framework that derived from MapReduce paper [[4]].

![Figure 1. Processing of MapReduce](image)

The MapReduce consists of map and reduce functions. These two functions are used as one unity in a job. Everything in MapReduce is consists of key value pair, either the input or output of the jobs. Each job consists of these two stages. First, a user-defined map function is applied to all input records to produce a list of key-value pairs. Second, a user-defined reduce function is called for each distinct or set key in the map output and do the processing associated with the key. There is also a combiner function that can act as a reducer in the mapper before the intermediate key is sent to appropriate reducer. Combiners are typically used to perform map-side "pre-aggregations" which aims to reduce the network traffic. Each mapper is assigned a portion of the input file called a split. By default, a split contains a single HDFS
block (64MB), so the total number of mapper is determined by the number of file blocks. The bigger the number of file blocks, the bigger the number of mapper. This will directly affects the processing time of MapReduce jobs.

B. Duplicate Stream Data Removal

As previously explained, the number of input file blocks greatly affects MapReduce processing time. In order to decrease the MapReduce processing time, we need to pre-process the input data for the mapper. In the context of stream data, such as RFID and sensor data, the data itself is a raw data obtain directly from RFID reader or sensor hub. The common types of errors in RFID or sensor data is redundancy readings (duplicate reads). This redundancy problem is recognized as a serious issue in RFID and sensor networks. Redundancy can happen at two different levels, reader level or data level.

- Reader level redundancy occurred when there are more than one readers or sensor hubs deployed to cover a specific location.

- Data level redundancy occurred as data streams. It happens when RFID tag or sensor tag stays at the same place for a period of time.

We are interested in duplicate stream data removal technique that considers location filtering. In [5], they use the set theory to solve a location and data duplication problems. Their methodology for location filtering is derived from a naïve set theory in which a set is described as a well-defined collection of objects. They proposed three algorithms. First, Intersection Algorithm, it compares EPC data between two readers. If the same data exists in both readers, this specific data will be allocated to the specific shelf. Second, Relative Complement Algorithm, it also compares EPC data between EPC readers. However, if there are duplications, these data will be ignored. The rest of the data will be allocated to the specific shelf. Third, Randomization Algorithm, this algorithm will randomize values between “0” and ‘1”. If the outcome is equal to “0”, the specific data will be allocated to the first reader and vice versa. But they only consider two readers in their algorithm, if we have more than two overlapping readers this algorithm won’t suffice.

Figure 2. Example for Duplicate RFID Data Elimination

In [6], they consider more than two overlapping readers in their algorithm. They proposed multi-level duplicate RFID reading filtering approach, local filtering and global filtering. Local filtering happens at the reader before the reading is sent to the middleware. The global filter at the middleware will filter duplicate readings among the readers before send it to the backend database or processing application. They refer the local filtering as Comparison Bloom Filter that uses only one filter.
3. Target Environment and Problem Definition

A. Target Environment

There are some wireless sensor network systems that monitor indoor and outdoor environmental condition using various sensors in the existing papers \([7]-[9]\). In \([7]\), some sensors measuring temperature, humidity were installed in vinyl greenhouse aimed at pear. In \([8]-[9]\), soil sensor, water sensor, temperature sensor, and humidity sensor were equipped in order to manage growth environment of farm products. In this way, most systems have installed sensors at fixed location.

![Figure 3. Constructing Sensor Network in a Cattle Ranch](image)

However, monitoring moving objects such as cattle, lamb, etc. needs a different construction of wireless sensor network, as shown in Figure 3. First, a ranch is divided at regular intervals and then a hub that has GPS and works to transmit the sensor data and GPS data to a server is installed in the center of the divided area. Each object attaches a sensor, which works to transmit the sensor data such as temperature and humidity to the hub periodically when one cattle enters the hub range. Because the sensor operates broadcast, both hubs receive the same sensor data in the overlapped area. Finally, the server uses Hadoop to store and to process the data for analysis.

B. Problem Definition

When a moving object such as cattle moves to another hub, the cattle will certainly pass an overlapped area between two hubs. As mentioned above, both hubs receive the same sensor data from the sensor of cattle in this time because of the characteristics of sensor. Because these duplication data cause to increase the intermediate data generated from map phase of MapReduce, then the number of mapper is increased and consequently operating time is increased.
The farmers place the hubs around the farm field. Since the cattle are an object that has its own mind, it moves and wanders around the field without being able to be controlled. Therefore, the farmers place several hubs overlapping with each other to make sure the whole farm field area is covered. This would make some reading data duplicated. When the cattle is in the overlapping area of some hubs, then this cattle’s sensor data would be read by these hubs, which is redundant. For example, we consider that a cattle moves from hub A to hub B (see Figure 4).

As described in Figure 5, if these data are used to map phase of MapReduce, and then the more increased input data, the more increased mapper. Consequently, operating time is increased.

4. Solution

To decrease the number of mappers, we try to prevent the duplication data to be processed during the map phase by proposing a duplication data removal technique. Firstly, we assign...
the hub ID with region number. We then find some duplication data by comparing the sensing values, hub ID, sensor ID and the timestamp parameters and introduce the removal technique.

A. Assign HubID

As mentioned above, the cattle ranch is divided into several patterned regions and a lot of hubs are installed in there.

![Figure 6. Assigning HubID](image)

When the cattle enter the overlapped area, we have to decide whether we are going to remove the data generated from original hub or destination hub. To map the region number, we define some rules as follows:

- The region number of left-side hub is smaller than right-side hub when viewed from above.
- The region number of upper-side hub is smaller than lower-side hub.

Figure 6 shows the implementation of the rules mentioned above. As we move to the right and down, the region number of hubID grows.

B. Search and Remove Duplication Data

Since the data transmitted by the hub includes some parameters such as hubID, sensorID, and timestamp, we try to compare these parameters among the data stored at the server for redundancy detection.

Given \( h_i \) is the region number of the hubs, when there are several reading data entries \( e \) which have the same sensor ID and their timestamp interval is less than 3 minutes (the reading cycle) between each other, then we classify them as intersection data and we will remove the redundant data. Based on the process described in Algorithm 1, since only single \( s^{th} \) entry with the smallest region number left, the data analytical processing would be more efficient and faster.
Algorithm 1. Duplication Data Removal Algorithm

```
Algorithm 1 shows the removal process of duplication data. With the incoming hubID (hID), timestamp (t) and sensorID (sID), the algorithm generates the result (b) whether the data is deleted or not. Among the duplicated data entries, we will keep only a single entry which has the smallest region number and delete all other entries.

As described in Error! Reference source not found., the cattle has moved from H00001 to H00002 and back to H00001 (Left hub -> Right hub -> Left hub). The parameters of line 6, line 10 and line 15 are almost the same except the timestamp. At this time, we can infer two results:

- The data of line 6 and line 10 is duplicate data because the difference between timestamp of line 6 and line 10 is 1 minute.
- Since the difference between timestamp of line 15 and line 6 or 10 is more than 3 minutes, so this data is not duplicate data.

Figure 7. Duplication Data and Not Duplication Data

5. Test

No published dataset for experimental purpose in terms of ranch management system can be found. Therefore, performance test evaluation is conducted using an ad-hoc application to generate the sensor network data as the dataset. Our performance measurements are made on
a Ubuntu personal computer with an Intel Core 2 Quad Q6600 (4 CPUs) 2.4 GHz processor, 4 GB of main memory, Java Runtime Environment 1.7 and Eclipse Indigo as development tool. Test dataset consists of hubID, time, sensorID, GPS, temperature and humidity, with the size of 1 and 2 GB. The first testing attempt is done by removing duplicates in a 1 GB data, resulting around 330 MB data. Second attempt is using a 2 GB data and the result is 670 MB data. Results show that the algorithm works well in minimizing the sensor network data size by removing the duplicates.

The next test is running the MapReduce process within a single node Hadoop cluster. This test is done by using all dataset, Original and Optimized, to demonstrate how data size will affect the processing time. Mapper will read each record from the dataset and then reducer will select a specific sensorID (specified by user) to be monitored. The result is as follows.

![Figure 8. Result of Performance Test](image)

In Figure 8, horizontal lines, which are the original data (1GB), takes longer time, 16 seconds, than the vertical lines, which are the optimized data (300MB), 10 seconds. Moreover, the original data (1GB) takes more than 22 seconds which is much longer than the optimized data (600MB), 13 seconds. This is because larger input file will require more mapper and thus take more time to be done. As the Hadoop file system will split the input data into 64 MB blocks of data, the bigger input size, the more blocks needed and the longer time taken before the map task finished. Therefore by minimizing the input size, the processing time will become shorter.

6. Conclusion

In this paper, we have proposed duplicate data removal technique, which help to remove duplicate or redundant data among huge amounts of data generated from sensor network when the cattle moves from a region of a certain hub to another one. Since finally the amounts of input data preprocessed are much smaller than the original data after the removal process, the number of mappers in MapReduce is decreased and eventually unnecessary processing time can be prevented.

Acknowledgements

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