Journal of Engineering and Applied Sciences 14 (13): 4367-4374, 2019
ISSN: 1816-949X

Data Summary Techniques based on MapReduce in Big Data

Jeong-Joon Kim
Department of Computer Engineering, Korea Polytechnic University,
Gyeonggi-do, 15073 Siheung-si, South Korea

Abstract: Wavelet, known as one of summary construction techniques was applied to feature extraction for multimedia data. Wavelet histogram is a summary technique which grafts wavelet on to histogram considered as a typical summary technique used in query optimization of database system and processing approximate query, etc. Wavelet histogram which combines merits of wavelet and histogram can generate a lossless optimal data summary of original data. In the existing studies, it needed more than one MapReduce job to construct local wavelet histogram of partial data stored in each node. In addition, it took a lot of time to construct the global wavelet histogram which is the combination of all local distributed wavelet histograms. Because the error bound for data reconstructed from wavelet histogram was not considered, there is a shortcoming that we cannot control the error of reconstructed data beforehand. In this thesis, we developed a wavelet histogram construction system which can construct wavelet histogram fast by one MapReduce job. Since, the error bound can beset before the construction of wavelet histogram, we can control the error of data reconstructed from wavelet histogram under the error bound. Finally, the efficiency of our wavelet histogram construction system was proved by comparing our system with others.

Key words: Data summary, MapReduce, big data, histogram, efficiency, reconstructed

INTRODUCTION

Histograms have been used to generate efficient summary information in many commercial databases and data management applications (Borthakur, 2007; Sklar, 1998; Dean and Ghemawat, 2004). The histogram is a technique for expressing the distribution of items of original data by dividing the number of items included in original data into a sub-area called a bucket. In addition, the wavelet histogram among various histogram methods has been studied recently because it has a small storage space and can be calculated in a short time (Garofalakis and Gibbons, 2004; Jestes *et al*., 2011). The wavelet histogram is a technique that uses wavelets successfully used in image/signal processing. Wavelet is a technique that converts data into wavelet coefficient which is summary information through wavelet transform (Muthukrishnan, 2005; Pang *et al*., 2013). And the wavelet histogram selects and stores only a part of the wavelet coefficients to obtain an additional summation effect.

MapReduce is a platform for distributed processing. It is used as a platform for data processing by Google and has recently attracted attention in various fields due to its excellent scalability and stability. Therefore, various algorithms developed in a single node environment are being reconfigured to run on this MapReduce platform. Conventionally, a system that generates a wavelet-based wavelet histogram has a slower generation rate because it goes through one or more MapReduce tasks. In addition, since, the tolerance of the error for the reconstructed data from the wavelet histogram can not be considered, the error of the reconstructed data can exceed the error boundary (Vitter and Wang, 1999; Shin *et al*., 2016).

However, the wavelet histogram generation system developed in this study consists of a data preprocessing manager, a histogram generation manager and a wavelet histogram generation manager and it is possible to generate a wavelet histogram in a short time through a single mapping task. In addition, the wavelet histogram generation system developed in this study can generate the wavelet histogram by matching the user defined error bounds, so that, the error of the restored data in the wavelet histogram can be limited within the error bounds. Finally, we verified the efficiency of the wavelet histogram generation system developed in this study through performance evaluation.

Literature review: This chapter describes the commonly used histogram and wavelet histogram techniques for summarizing data, analyzing Hadoop platform HDFS and MapReduce and explaining previous studies that generate data summaries on the Hadoop platform.

Histogram: Histograms are one of the techniques to effectively summarize original data and are widely used for

Table 1: Example of creating a Equi-width histogram

Bucket = (Start point, end point)	Item belonging to	No. of items	(Bucket, number of items)
B1 = (1, 2)	1.61, 1.72	2	(B1, 2)
B2 = (2, 3)	2.23, 2.33, 2.71, 2.90	4	(B2, 4)
B3 = (3, 4)	3.41	1	(B3, 1)
B4 = (4, 5)	4.21, 4.70, 4.82, 4.85, 4.91	5	(B4, 5)

measuring selectivity (Dean and Ghemawat, 2004; Jestes *et al*., 2011). The selectivity estimation means that the number of data items belonging to a certain range query is estimated. The histogram technique is very popular in commercial database systems because it uses very little space and does not use information about the distribution of data to summarize. In addition, the histogram is a data summarizing technique that divides the original data into buckets to indicate the degree of distribution of the original data. The histogram is a set of buckets and each bucket stores information on the number of original data items (frequency) existing in the bucket. A bucket has a starting point and an ending point and the distance between the starting point and the ending point indicates the length of the bucket. Equi-width histogram has a feature that is easy to generate all the same bucket width. The process of creating the Equi-width histogram is described below as an example:

$$D = \{1, 61, 1.72, 2.23, 2.33, 2.71, 2.90, 3.41, 4.21, 4.70, 4.82, 4.85, 4.91\}$$

When the above original Data D exists, the domain of the original data D is (Borthakur, 1998; Muthukrishnan, 2005). If the original data D is divided by B1-B4 and the bucket length is set to 1 an Equi-width histogram can be generated as shown below. Table 1 shows an example of generating an Equi-width histogram.

Since, the bucket length is set to 1 as in Table 1, the original data can be divided into 4 buckets from (Borthakur, 2007; Dean and Ghemawat, 2004) to (Jestes *et al*., 2011; Muthukrishnan, 2005). After the number of occurrences of items belonging to each bucket is counted, the original Data D can be converted into the form of (bucket, item number). In addition, B1-B4 buckets represent four buckets. Figure 1 is a graphical representation of the Equi-width histogram.

As shown in Fig. 1, the items in original data D mainly belong to Bucket B2 and Bucket B4. In other words, the items of the original Data D are concentrated mainly on B2 = (2, 3) and B4 = (4, 5). Using Equi-width histogram generated in Table 1, it shows the estimation of the selectivity for the range query to output the number of data items belonging to (1.1, 4.5). First, the number of items of original data belonging to (1.1, 4.5) is composed of the following two parts:

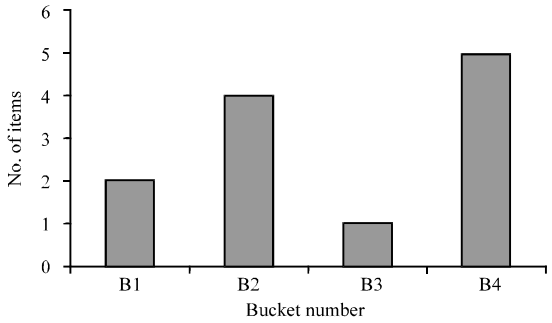

Fig. 1: Graph representation of Equi width histogram

- The number of items in the original data belonging to (1.1, 2) and (4, 4.5)
- The number of items in the original data belonging to (2, 4)

Since, there is no bucket corresponding to (1.1, 2) in case 1, the number of items of original data belonging to (1.1, 2) is calculated as the ratio of the number of items of original data belonging to Bucket B1 = (1, 2). By applying the following formula, the number of data items belonging to (1.1, 2) can be estimated:

$$\text{Estimate} = \frac{\text{End point of bucket} - \text{Start point value of estimated range}}{\text{Bucket endpoint value} - \text{Bucket start point value}} \quad (1)$$

When Eq. 1 is applied, the estimated value of the number of items of original data belonging to (1.1, 2) is (2-1.1)/(2-10) * 2 = 1.8. Similarly, the estimated value of the number of items of original data belonging to (4, 4.5) is [(4.5-4) / (5-4)] * 5 = 2.5. In the case (2), the bucket corresponding to (2, 4) is Bucket B2 = (2, 3) and B3 = (3, 4). Therefore, the number of items of original data belonging to (2, 4) is 4+1 = 5. When, the calculation results of 1 and 2 are combined, the number of items of original data belonging to (1.1, 4.5) is 1.8+2.5+5 = 9.3. The result is that the error between the correct answer and 8 is as small as |9.3-8| = 1.3. However, there may be a large error in estimating the number of data items with respect to original data in which a particular data item frequently appears. Assuming that the number of items in Bucket B4 in Table 1 is 100, the estimated number of original data items belonging to (4, 4.5) is [(4.5-4) / (5-4)] * 100 = 50

and the number of items of original data belonging to (1.1, 4.5) is 1.8+50+5 = 56.8. The result is very large as |56.8-8| = 48.8 with the correct answer of 8.

MATERIALS AND METHODS

Wavelet: Wavelet is a representative data summarization technique that has been used for query optimization and rough query processing of database systems (3, 6). The wavelet summarizes the original data by transforming it into a wavelet coefficient through a wavelet transform technique. Assuming that the same storage space is used, the wavelet has a small error when summarizing the original data in which a particular data item frequently appears as compared to the Equi-width histogram. Wavelet transform is one of the wavelet transform techniques used in wavelet transform (HaW Wavelet Transform, HWT). The wavelet transform scheme transforms the data to be summarized into wavelet coefficients and the wavelet coefficients are composed of average coefficients and detail coefficients. The calculation procedure of average coefficient and detail coefficient is explained as follows, for example.

Number of items in the data set in D:

V = {3, 5, 10, 8, 2, 2, 10, 14}

There is one dataset D, v = {v (1), ..., V (x)} is an array of the number of items appearing in D, it is possible to convert "AAAAA" into a wavelet coefficient through a wavelet transform technique as a summation object. Figure 2 is the wavelet coefficient tree used to calculate the wavelet coefficients. As shown in Fig. 2, the wavelet coefficient tree consists of nodes v (1)~v (8) and nodes w_0~w_7. The nodes v (1)-v (8) are the number of data items v has. The nodes w_0-w_7 have low wavelet coefficient values and the low wavelet coefficients are composed of one average coefficient w_0 and seven detailed coefficients w_1-w_7. The average coefficient w_0 is an average value of v (1)-v (8) and the detailed coefficient is divided by level 1~level 3 and calculated by the order of level 3-> level 2->level 1. The detail coefficient can be calculated using the following Eq. 2:

$$\text{Detailed coefficient} = \frac{\text{Right subtree sum} - \text{Left subtree sum}}{\text{The number of nodes in the subtree}} \quad (2)$$

The process of calculating the detailed coefficient w_4 using the Eq. 2 will be described as follows. The nodes belonging to the left / right subtree of the node with w_4 are v (1) and v (2), respectively. The right subtree sum is 5, the left subtree sum is 3 and the number of nodes in the subtree is 2. Therefore, the detailed coefficient w_4 = (5-3)/2 = 1. Another example is that v (1) and v (2) belong to the left subtree of the node with the detail factor w_2, v (3) and v (4). The number of nodes is 4. When the Eq. 2 is applied, the detailed coefficient w_2 = ((10+8)-(3+5))/4 = 2.5. The remaining detailed coefficients can also be calculated through Eq. 2.

In addition, the values of v (1)-v (8) can be restored by using the tree in Fig. 2. Equation 3 is a formula for restoring the values of v (1)-v (8) using HaW wavelet coefficients:

$$D(i) = \sum_{j \in path\ (i)} sign(i, j) \times w_j$$
$$sign(i, j) = \begin{cases} 1,\ j = 0 \text{ or } \upsilon(i) \in \text{left subtree of } w_j \\ -1,\ \upsilon(i) \in \text{right subtree of } w_j \end{cases} \quad (3)$$

D (i) is the restored value of υ(i) as shown in Eq. 1. w_j is a wavelet coefficient node value and sign (i, j) is a variable indicating whether w_j is positive or negative. And path (I) refers to the route from one leaf node to the root

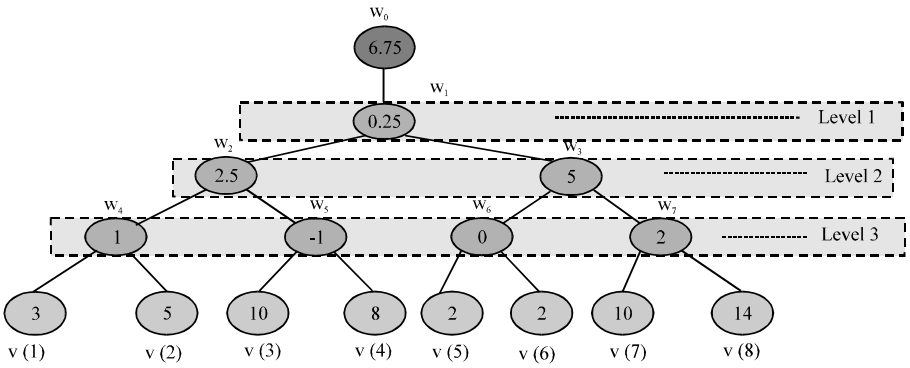

Fig. 2: HaW wavelet transform coefficients tree

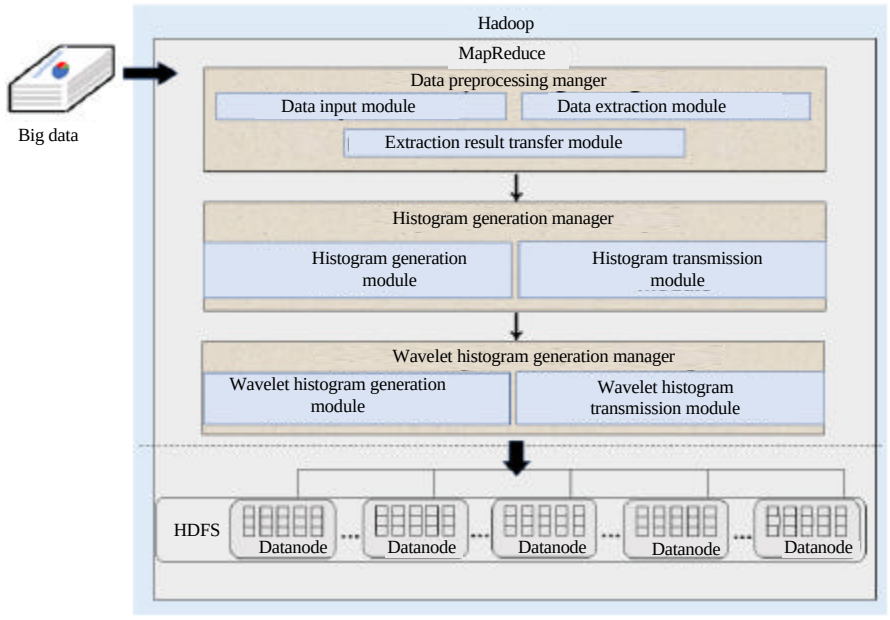

Fig. 3: System overall structure

node with reference to Fig. 2. When restoring data, sign (i, j) = 1 when starting from the root node and $\upsilon(i)$ belonging to the left subtree of w_j and sign (i, j) = -1 to be. Therefore, it is determined by $\sum_{j\in \text{path}(i)} \text{sign}(i, j)\times w_j$ of D (i)

System design

Structure of the entire system: An efficient wavelet histogram generation system based on MapReduce developed in this study provides a RecordReader that reads the contents of data according to the data storage format. A wavelet histogram with less storage space and without exceeding the error boundaries can be generated through a single mapping task.

We have designed an efficient wavelet histogram generation system based on MapReduce by applying the functions described above. Figure 3 is a schematic diagram of an efficient wavelet histogram generation system based on MapReduce.

An efficient wavelet histogram generation system based on MapReduce consists of a data preprocessing manager, a histogram manager and a wavelet histogram manager.

The data preprocessing manager consists of a data input module, a data extraction module and an extraction result transmission module. After reading data from a RecordReader corresponding to the type of big data, it extracts the value possessed by the attribute and transmits the extracted result.

The histogram generation manager consists of a histogram generation module and a histogram transmission module. It converts the attribute value received from the data preprocessing m a nager into a pair of key and value and outputs and transmits the attribute value.

The wavelet histogram generation manager consists of a wavelet histogram generation module and a wavelet histogram storage module. Histogram generation It is responsible for generating a pair of key and value received from the manager as a wavelet histogram that does not exceed the error boundary.

Preprocessing data, administrator: The data preprocessor extracts the attribute values to be summarized from the attribute values of the big data to be analyzed.

Data input module: In order to save storage space, big data is often compressed and stored and the storage format of data is also various. Therefore, the data input module reads the big data of various storage formats using the corresponding RecordReader

Data extraction module: The data extraction module is responsible for extracting the corresponding attribute value from the data read from the data input module.

Extraction result transfer module: The extraction result transmission module receives the result extracted from the data parsing module and transmits the result to the histogram generation module of the histogram generation manager.

Histogram generation, administrator: The histogram generation manager consists of a histogram generation module and a histogram transmission module. It is responsible for converting the attribute values received from the data preprocessing manager into a histogram by using a mapper deuce mapper.

Histogram generation module: The histogram consists of a pair of key and value. The key is the only value that appears in the received attribute value and value is the number of times the attribute value appears.

Histogram transmission module: The histogram transmission module receives the local histograms generated by the histogram generation module and transmits the received local histograms to the wavelet histogram generation manager.

Wavelet histogram generation m anager: The wavelet histogram generation manager is composed of a wavelet histogram generation module, a wavelet histogram storage module and a restoration module. The histogram generation manager generates a wavelet histogram that does not exceed an error boundary using a MapRedeuce reducer.

Wavelet histogram generation module: The wavelet histogram generation module receives and merges local histograms composed of (Key, value) pairs from the histogram generation manager. Also, the values of the same key are merged and the merged (Key, value) pair constitutes a global histogram.

Wavelet histogram storage module: The wavelet histogram storage module stores the wavelet coefficients generated in the wavelet histogram generation module and all the keys included in the global histogram in HDFS. The reason for storing the key is that it must be restored as a key and value pair when restoring data from the wavelet histogram later.

RESULTS AND DISCUSSION

Implementing system: This chapter describes the manager implementation of the efficient wavelet histogram generation system based on MapReduce. Data preprocessing manager, histogram generation manager and wavelet histogram generation if you create a jar file that contains the functions of the administrator and give that option to the jar file that option performs the corresponding function.

Preprocessing of data, administrator: The data preprocessor is responsible for extracting the attribute values from the big data to be analyzed. An example of extracting attribute values in Fig. 4.

As shown in Fig. 4, the attribute value is extracted from the contents of the hdfs://hadoop:9000/input/ test.dat file using the command to extract the attribute value and the file name is stored in the hdfs://hadoop:9000/home/output/ as parse result. The extracted attribute values are shown in Fig. 5.

As shown in Fig. 5, the content of the test.dat file is a record of a user's visit to a server and includes six attributes (UserID, time, method, resource, response, size). These six attributes indicate the user ID, connection time, connection method, visited data path, response status code and response content size, respectively which are connected in order. In order to estimate the distribution of users connected to the server, the value of the UserID attribute which is the user ID to access the server is extracted and stored in the parseresult.dat file.

Histogram generation manager: The data preprocessing manager converts the extracted data into a histogram composed of (Key, value) and transmits it to the wavelet histogram generation manager. An example of a command to generate a histogram is shown in Fig. 6.

Convert the extracted attribute values in the hdfs://hadoop:9000/input/parseresult.dat file into a histogram composed of (Key, value) using the command to generate the histogram as shown in Fig. 7 and save the file name as histogram.dat in hdfs://hadoop: 9000/home/output. The contents of the histogram.dat file can be found in Fig. 7.

As shown in Fig. 7, the histogram.dat file that stores the histogram information is composed of two columns of data such as ① and ②. The ① column is the UserID value that exists in the extracted attribute value uniquely and the ② column is the number of times the UserID value appears.

Wavelet histogram generation, administrator: The wavelet histogram generation manager receives the histogram from the histogram generation manager and generates a wavelet histogram satisfying the error bounds. Figure 8 is an example of a command to generate a wavelet histogram.

As shown in Fig. 8, the histogram in the file hdfs://hadoop: 9000/input/histogram.dat is converted to the wavelet histogram with the error boundary 10 using the command to generate the wavelet histogram and the file is saved as wh10 in hdfs://hadoop:9000/output/. The contents of the wh10 file can be found in Fig. 9.

```
hadoop@ubuntu:~$ hadoop jar wavelethistogram.jar -p 1 hdfs://hadoop:9000/h
ome/input/test.dat hdfs://hadoop:9000/home/output/ parseResult.dat
```

Fig. 4: Example of command to extract attribute value

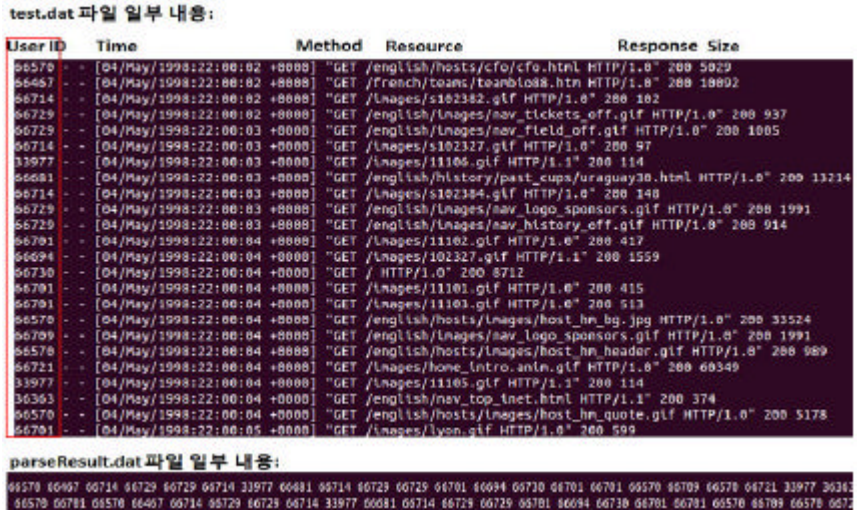

Fig. 5: Example of extracting attribute values from data preprocessing manager

```
hadoop@ubuntu:~$ hadoop jar wavelethistogram.jar -c hdfs://hadoop:9000/hom
e/input/parseResult.dat hdfs://hadoop:9000/home/output/ histogram.dat
```

Fig. 6: Example of command to generate histogram

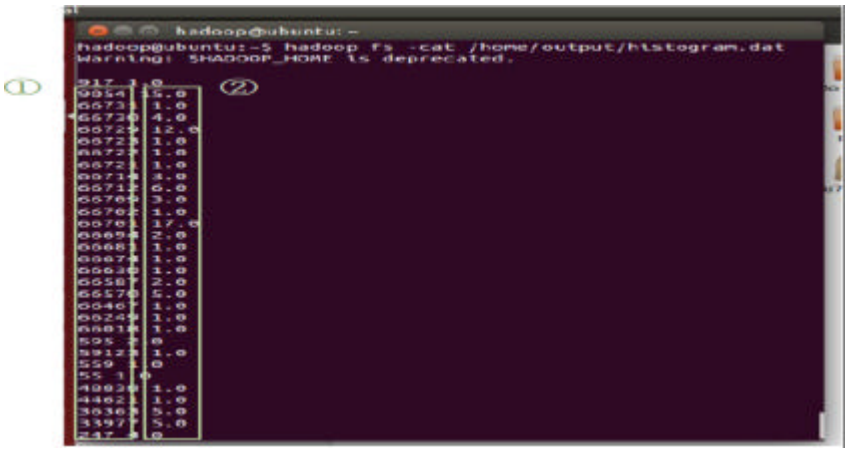

Fig. 7: Screen to look up contents of histogram.dat file

```
hadoop@ubuntu:~$ hadoop jar wavelethistogram.jar hdfs://ubuntu:9000/home/input/h
istogram.dat hdfs://ubuntu:9000/home/output/ wh10 10
```

Fig. 8: Example of command to generate wavelet histogram

As shown in Fig. 9, the wh10.dat file in which information related to the wavelet histogram is stored is composed of data in two columns such as ① and ②. The ① column is the index indicating the position of the wavelet histogram coefficient and the ② column is the wavelet histogram coefficient.

Data can be restored from the wavelet histogram by executing the data restoration command. Figure 10 is a command to restore the data from the wavelet histogram.

As shown in Fig. 10, restoration of wavelet histogram data in hdfs: //ubuntu: 9000/ home/ input/ Wavelet

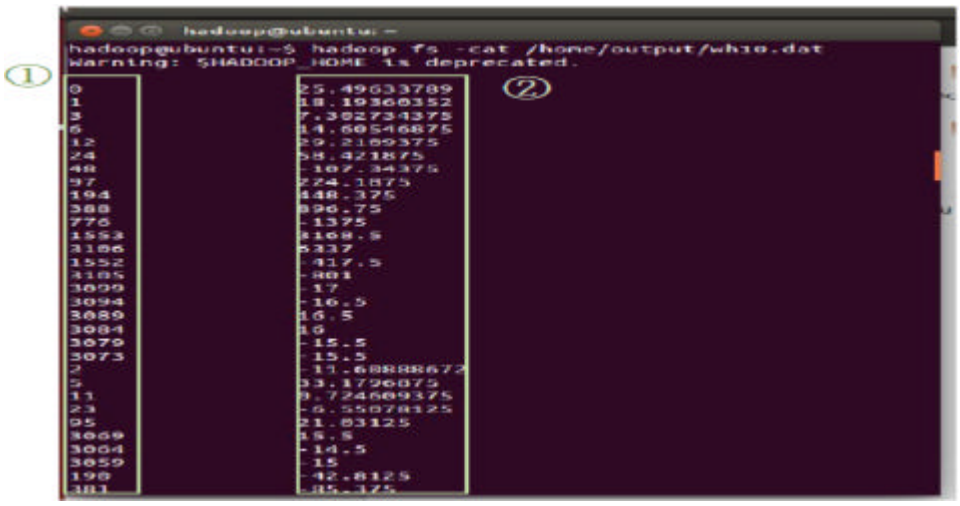

Fig. 9: Some contents of the wh10.dat file

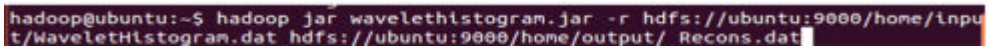

Fig. 10: Example of command to restore data

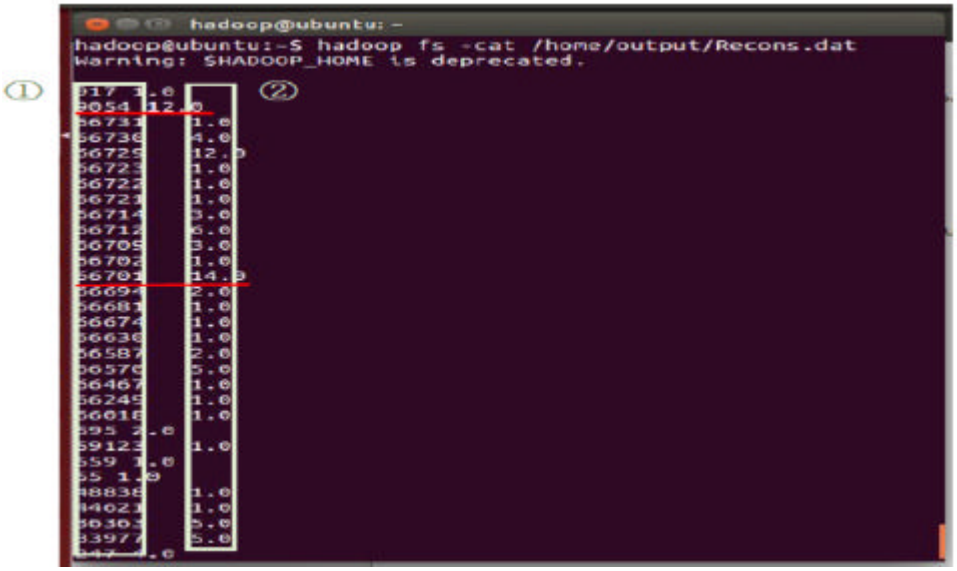

Fig. 11: Some contents of the Recons.dat file

Histogram.dat file is saved and saved as file name Recons.dat in hdfs://ubuntu:9000/home/output/. The contents of Recons.dat can be seen in Fig. 11.

As shown in Fig. 11, the Recons.dat file in which the data restored from the wavelet histogram is stored is composed of data in two columns as in ① and ②. The ① column is the UserID value and the ② column is the number of restored data items. For example, UserID which is 66730 has 4 occurrences. When, the number of items of UserID 9054 and 66701 is compared with the number of items of UserID 9054 and 66701 in Fig. 11, there are 3 errors but it does not exceed 10 error boundaries.

CONCLUSION

Since, big data has a large amount, fast creation speed and various properties, it is more efficient to process big data summary information than big data itself. The wavelet

histogram which is one of the summary information generation techniques can generate optimal data summary information that does not cause loss of information of original data. MapReduce is being used as a framework for Google's data processing and has recently gained popularity in a variety of areas due to its excellent scalability and stability. Therefore, efficient wavelet histogram generation system is required by using MapReduce.

In this study, we have developed a wavelet histogram generation system in a distributed MapReduce environment. The wavelet histogram generation system developed in this study can generate the wavelet histogram through only one mapping task, so, the entire wavelet histogram generation time is reduced. In addition, since, the user can specify an error boundary in advance before generating the wavelet histogram, the error of the restored data from the wavelet histogram can be adjusted in advance.

ACKNOWLEDGEMENT

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2017R1A2B4011243).

REFERENCES

Borthakur, D., 2007. The hadoop distributed file system: Architecture and design. Apache Software Foundation, Forest Hill, Maryland, USA.

Dean, J. and S. Ghemawat, 2004. MapReduce: Simplified data processing on large clusters. Proceedings of the 6th Symposium on Operating Systems Design and Implementation, December 6-8, 2004, San Francisco, CA., USA., pp: 137-150.

Garofalakis, M. and P.B. Gibbons, 2004. Probabilistic wavelet synopses. ACM. Trans. Database Syst., 29: 43-90.

Jestes, J., K. Yi and F. Li, 2011. Building wavelet histograms on large data in MapReduce. Proc. VLDB. Endowment, 5: 109-120.

Muthukrishnan, S., 2005. Subquadratic algorithms for workload-aware haar wavelet synopses. Proceedings of the International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2005), December 15-18, 2005, Springer, Berlin, Heidelberg, Germany, ISBN:978-3-540-30495-1, pp: 285-296.

Pang, C., Q. Zhang, X. Zhou, D. Hansen and S. Wang *et al*., 2013. Computing unrestricted synopses under maximum error bound. Algorithmica, 65: 1-42.

Shin, I.S., J.J. Kim, Y.S. Lee and J.Y. Moon, 2016. NoSQL-based spatial data processing systems in big data environments. Intl. Inf. Instit. Tokyo Inf., 19: 4219-4236.

Sklar, B., 1998. Digital Communications. Prentice Hall, New Jersey, USA., Pages: 187.

Vitter, J.S. and M. Wang, 1999. Approximate computation of multidimensional aggregates of sparse data using wavelets. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '99) Vol. 28, May 31-June 03, 1999, ACM, New York, USA., pp: 193-204.