International Journal of Innovative Computing, Information and Control
Volume **14**, Number **2**, April **2018**

ICIC International ©2018 ISSN 1349-4198
pp. **603–614**

A DATA BALANCE ALGORITHM BASED ON CONTENT SAMPLING HISTOGRAM IN MAPREDUCE

Zhe Wei[1], Honglin Zeng[2] and Weibo Zhou[3]

[1]School of Computer Science
Civil Aviation Flight University of China
No. 46, Nanchang Road, Guanghan 618307, P. R. China

[2]School of Mechanical Engineering
Southwest Petroleum University
No. 8, Xindu Road, Chengdu 610500, P. R. China

[3]Chengdu Chongxin Big Data Services Ltd.
Chongzhou 611200, P. R. China
findz@qq.com

Received April 2017; revised August 2017

Abstract. *MapReduce that is widely used in the fast processing of massive dataset is a typical distributed computing model and its performance largely depends on the state of data distribution. Because the data content tends to be uneven together with the randomness of the storage, data skew often occurs during the computing in MapReduce. The study on the data skew in MapReduce currently mainly centers on the Reduce side and less on the Map side. This paper extracts the data features by the data sampling, constructs the histogram of data blocks in the Map side, judges the data skew degree of each storage node according to the distribution status of data blocks, defines the file balance deviation as the metrics of data skew of the entire file, and reduces the file balance deviation by the data balancing algorithm. This paper proposes a content based data balance algorithm which adopts the greedy strategy and can achieve a better approximate solution to the optimal solution of the data balance distribution in the Map side. Through several experiments with different data sets and compared with the random data block distribution algorithm, the proposed algorithm can significantly reduce the deviation of file balance and offers a better data balancing effect.*

Keywords: Sampling, Data content, Histogram, Block, Data balance

1. Introduction.

With the speeding up of data generation, the data scale that needs to be processed from all walks of life expands dramatically. In the face of mass data processing requirements, big data technology has set off a worldwide study and all kinds of big data processing systems get great development. The open source Hadoop [1] from Apache with high reliability and low cost advantage quickly becomes a widely adopted distributed solution in the field. Hadoop consists of two key parts: HDFS [2] and MapReduce [3]. HDFS is used to solve the problem of mass data storage while MapReduce aims to deal with the parallel computing of mass data. MapReduce has proven to be an efficient data processing model but the drawback is that its performance is greatly influenced by the data distribution state [4] and the appearance of data skew will cause a (or some) task to take more time than the others resulting in the bottleneck of the whole system. Thus, the study on the data distribution state and the data skew problem has important significance to improve the MapReduce performance.

The main contribution of this research is to propose a content based data balance algorithm. It first describes the data distribution state in MapReduce by means of sampling and next judges whether there exists data skew problem. By using the greedy strategy [5] and through the method of data block exchange, the proposed method balances the data distribution without changing the storage capacity of each storage node; therefore, it can achieve a better approximate solution to the optimal solution of the data balance distribution in the Map side.

The organization of this paper is as follows: related work is presented in Section 2; preliminaries and the proposed method are described in Section 3 and Section 4 respectively; experimental tests and result analysis are shown in Section 5; Section 6 concludes this work.

2. Related Work.

The histogram is often used to describe the state of data distribution, which lays the foundation for the efficiency optimization in the distributed parallel computing [6]. By using the tuple sampling method, [7] constructed the wavelet histogram based on MapReduce. [8] proposed a Maxdiff histogram construction method based on MapReduce. The framework of MapReduce is extended in [9] by adding data sample and statistics before Map and after Reduce; thus, the construction method for the same width and depth histogram based on MapReduce is improved.

The computing process of MapReduce consists of two phases, i.e., Map and Reduce. The parallel process of data is implemented in the Map phase and the gathering process of data is carried out in the Reduce phase [10]. Users only need to use the Map() function and Reduce() function to implement the service logic and complete the distributed computing. The map() function reads and writes data in the form of $< key, value >$. These data are converted into other $< key, value >$ pairs and they are written into the disk as middle data. The Reduce() function reads and writes the $< key, value >$ pairs from the Map() function through the process of Shuffle. The pairs of the same Key are merged and new $< key, value >$ pairs are generated. The final results are output to the HDFS.

The bottleneck of performance may occur in both the Map phase and the Reduce phase. The reasons that data skew may be generated in the Map phase and Reduce phase are comprehensively analyzed in [11-14] and it is indicated that the imbalance distribution of raw data will result in one or some Map tasks spending more time in processing the same amount of data than others, and it is also indicated that the skew in the data content itself and unreasonable default Hash partitioning method in Hadoop lead to one or some Reduce tasks having far more amount of data to be processed than others.

In the Map phase, data is often treated without being preprocessed, so the data distribution state cannot be obtained. In addition, the data skew in the Map phase is closely related to the data content, but the data content is the attribute of the data itself and it cannot be changed. Therefore, effective balance cannot be carried out in the Map function. In the Reduce phase, data has already been processed in the Map phase and the data distribution state has been obtained; thus, the balance treatment can be done in the Reduce function. Few studies are now centered on the data balance in the Map phase while most focus on the data balance in the Reduce phase. The sampling strategy based on tuple is proposed in [15,16]. The data block based sampling strategy is proposed in [17,18] which first predict the data distribution and then adjust the partition function. An incremental partition strategy is proposed in [19] which implements the data balance in the Reduce side by multiple rounds of increment strategy.

Based on the above research status, this paper tries to preprocess the data by the sampling method, based on which the data content based histogram is constructed to implement the measurement and judgement on the data skew status in the data blocks and the whole file. The raw data of the whole file is balanced through the data balance algorithm which makes each node in the Map side as balanced as possible.

3. Preliminaries.

3.1. Data sampling.

Data skew in the Map side is mainly resulted from the imbalanced distribution of the input data. If the state of data distribution can be known, it can be adjusted and distributed reasonably and the data skew can be reduced. Because MapReduce is targeted for the large-scale data processing, if the distribution state of the whole data is to be analyzed, the cost would be too high. Thus, the distribution of data can be predicted in the manner of sampling. It can be proved that the sampling data itself is the result of multiple Bernoulli trials; therefore, it must obey the binomial distribution, and when the number of tests is high enough, the binomial distribution approximately obeys the normal distribution. Obviously, the more the sampling data is, the higher the accuracy of the data distribution becomes, and the higher the sampling cost is. In this article, the statistics information about the frequency of Key is obtained by adding a sampling task. The details are as follows.

Step 1: the sampling size is determined by the Split size and the sampling coefficient s, e.g., SplitSize = 128 MB, $s = 5\%$, then SamSize = $128 * 5\% = 6.4$ MB;

Step 2: set N sampling partitions to divide Split into N partitions with the aim to eliminate the influence of data locality principle on the random sampling, e.g., $N = 3$;

Step 3: randomly sample SamSize/N data in every partition;

Step 4: compute the sum of frequencies of Key that appears in every Split sampling data;

Step 5: count the sum of frequencies of Key in all the sampling data in the Reduce side.

The data sampling algorithm is presented in Table 1. The sampling coefficient s is related to the accuracy of the calculation results, and its optimal value should be determined several times according to the accuracy requirement and the computational power. The purpose of setting up partition N is to exclude the influence of the principle of data locality on the random sampling. It is suggested that in the sampling process, its optimum value should be determined several times according to the size of data block and the distribution of data, and 3 to 5 is recommended for N.

TABLE 1. Data sampling algorithm

Algorithm

```
Input: sampling coefficient $s$ and the number of partitions $N$
Output: sum of frequencies of Key that appears in every Split sampling data
1) SamSize = SplitSize $* s$
2) SamData[$N$][SamSize/$N$]
3) $i = 0$
4) while $i < N$
5) SamData[$i$] [SamSize/$N$] $\leftarrow$ random sampling
6) $i++$
7) end while
```

3.2. Histogram of sampling data.

According to the key status in the sampling data, all keys are divided into KeyGNum groups, based on which the histogram of data blocks and files are constructed.

Definition 3.1. *Suppose the key value in the data sampling is x, then the frequency sum of each subgroup is defined by*

$$y = f(x),\ x \in KeyG_i,\ i \in [1, KeyGNum]$$

Definition 3.2. *Set $keyG_i$ as X-axis and $f(x)$ as Y-axis, construct histogram H and its mathematical expression is:*

$$H = \{< keyG_i, f(x) >,\ i \in [1, KeyGNum],\ x \in KeyG_i\}$$

Definition 3.3. *Suppose the original data is divided into n Blocks, every Block is divided into m key subgroups when the histogram is constructed, then all the data histogram information of Block in this file can be expressed by the following matrix:*

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$$

where the ith line indicates the data histogram information of the ith Block, the jth column represents the jth key subgroup in the histogram, and a_{ij} denotes the frequency information of the ith Block in the jth key subgroup.

Because the data file uses fixed partition size when it is divided into Blocks, if the end of the Block differences can be ignored, the following constraint condition can be obtained.

$$\sum_{k=1}^{k=m} a_{ik} = \sum_{k=1}^{k=m} a_{jk}, \quad (i \neq j) \tag{1}$$

The above matrix A and the constraint condition can be treated as a mathematical expression of file histogram based on data content.

If n Blocks are distributed and stored on N storage nodes ($n \gg N$) without changing the storage capacity of each node and the frequency sum of each subgroup tries to achieve a balanced distribution, the problem is transformed into dividing n row vectors into N shares and making the values of m components in each share tend to be equal in matrix A that contains constraint condition. The nature of the problem is to find a global optimal solution of balanced distribution of a block without changing the data content and the number of blocks of each node.

4. Content Based Data Balance Algorithm.

The sampling data histogram can approximate to describe the distribution status of the original file on each storage node in a distributed system. According to the distribution, the related mathematical models can be defined.

4.1. Related definitions.

If n blocks are equally allocated to N nodes, some nodes may have more Blocks and others may have less Blocks. Obviously under the condition of the same MapTask, nodes with more Blocks will have more computing time overhead than those with less Blocks.

Definition 4.1. (Node balanced vectors). *Suppose the maximum number of Blocks that each node has is k, then node balanced vectors can be defined as*

$$A_{avg} = (av_1\, av_2 \ldots av_m) = \frac{\sum_{i=1}^{n} a_{i1}\ \sum_{i=1}^{n} a_{i2}\ \sum_{i=1}^{n} a_{im}}{n} * k$$

The arbitrary k row vectors in matrix A are added and combined into a new row vector, the newly constructed matrix is defined as A_k, and then A_k must have C_n^k row vectors.

There are many similarity measurements between the row vectors and the balanced vectors in A_k, e.g., Euclidean distance, Manhattan distance, Markov distance, and Angle cosine. Considering the proposed algorithm focusing on whether the frequency of related components on each storage node is equal or not and placing more weight on the numerical values, the Euclidean distance is selected as measurement for the similarity between vectors in the proposed method.

Definition 4.2. (Combined matrix A_k). *Suppose p is a row vector and $p = (a_{i1}\, a_{i2} \dots a_{im})$, the Euclidean distance between p and balance vector is*

$$d = \sqrt{(a_{i1} - a_{v1})^2 + (a_{i2} - a_{v2})^2 + \ldots + (a_{im} - a_{vm})^2}$$

Definition 4.3. (File balancing deviation fd). *Add all the distance between the row vectors (constructed by all the nodes that have k blocks) and the balance vectors, define the sum as the file balancing deviation, and suppose there are N_k nodes that have k Blocks.*

$$fd = d_1 + d_2 + \cdots + d_{N_k}$$

4.2. **Algorithm description.** The proposed data balance algorithm achieves the content balance of every node by exchanging the Block without changing the number of Blocks of each node. Thus to determine how many Blocks are stored on each node is the premise of the proposed algorithm. According to the histogram information and Block number distribution information, this paper proposes the data balance algorithm based on data histogram. The details are as follows.

Step 1: Compute the node balancing vectors;

Step 2: Construct the combined matrix A_k;

Suppose the max Block number is k in the Block distribution information table, the combined matrix A_k needs to be constructed. For example, a file is distributed on 5 nodes and the block distribution on each node is presented in Table 2, it can be noticed that the max Block number is 3, and then the combined matrix needed to be constructed is A_3 which has $C_{14}^3 = 364$ row vectors.

TABLE 2. Block distribution of a file

	Node1	Node2	Node3	Node4	Node5
Block	3	3	3	3	2

Step 3: Compute vector distance;

Compute the distance between all the row vectors in A_k and the node balanced vectors.

Step 4: Allocate Blocks according to the distance;

Select N unrelated row vectors with minimum d value from A_k, and allocate several Blocks that construct the row vector to the same storage node.

The data balance algorithm is shown in Table 3.

5. **Experimental Tests.**

5.1. **Settings.** In order to verify the algorithm validity and superiority, we use the virtual machine to build a Hadoop cluster composed of 6 nodes to carry out the test. The Hadoop cluster has 2 NameNode nodes and 4 DataNode nodes and each node has 512 MB memory and 8 GB hard drive. The OS is CentOS 6.5 and the Hadoop version is 2.6.0. Further, the values of sampling coefficient s and the values of sampling partition N are set based on several repeated experiments in order to highlight the experimental contrast effect. Their values can be adjusted with almost no influence over the test results.

Step 1: The sampling algorithm is used to construct the data histogram of the entire file. The 240 billion score data set collected by the GroupLens project team in the college of computer science from the university of Minnesota is stored to the Hadoop cluster constructed by the test, set the sampling coefficient s as 5%, 20% and 100%, set the

Table 3. Data balance algorithm

```
Algorithm
Input: Matrix $A$ and Block distribution quantity
Output: Block distribution information
1)  $A_{avg}$[group] $\leftarrow$ $A[n]$[group];
2)  while $A$ is not null and $n > k$
3)  $A_k[i]$[group] $\leftarrow$ $A[n]$[group]; // Take any $k$ rows data from matrix $A$
                                  // accumulate into one row into matrix $A_k$
4)  Index$[i][k]$ $\leftarrow$ $1 \ldots k$; // Mark the $k$ rows subscript composed of $A_k[i]$[group]
                           from matrix $A$
5)  $d_i \leftarrow |A_k[i]$[group] $- A_{avg}$[group]$|$;
6)  min $\leftarrow d_i$ // Search minimum value in $d_i$
7)  $A$[Index$[i][0]$][group] $\ldots A$[Index$[i][k]$][group]; //Find $k$ rows data from matrix $A$
8)  $A[n-k]$[group] $\leftarrow$ $A[n]$[group] // Delete the $k$ row data from matrix $A$
9)  $n = n - k$;
10)   end while
```

sampling partition N as 3, construct the data histogram of the data block and the data histogram of the entire file and analyze the status of the data skew.

Step 2: According to the related information of the data histogram of the entire file and the data histogram of every data Block, allocate the two schemes of using random distribution and data histogram based data balance algorithm, compare the file balancing deviation, and analyze the data skew status according to the node data histogram.

5.2. **Results and analysis.** The data set that is used in the test is the 24000000 score data of 40000 movies from 260000 users, the score range is $(0, 5)$, the data set size is 632.69 MB, data update time is in October 2016. Because the Block size is set to be 64 MB in the test clusters, the experimental data file is divided into 10 Blocks.

5.2.1. *Data histogram.* When $s = 5\%$, the subgroup, the frequency of each group and the statistics information in every Block are presented in Table 4.

Table 4. Sampling data I

	$[0,1)$	$[1,2)$	$[2,3)$	$[3,4)$	$[4,5]$
Block 1	6674	6376	29883	45014	34073
Block 2	4101	15935	21667	42892	37425
Block 3	16813	6993	49254	35180	13780
Block 4	3659	8595	17629	65541	26596
Block 5	13815	10240	27463	50131	20371
Block 6	4750	11825	51924	18450	35071
Block 7	8869	14358	30807	46770	21216
Block 8	5367	25892	7165	50514	33082
Block 9	5955	9562	42788	39454	24261
Block 10	16445	13819	46275	12116	33365

Use the sampling data information from the above table to construct the sampling data histogram of each Block which is presented in Figure 1.

All sampling data Block histogram information is combined on the Reduce node, and the histogram of the entire file sampling data is obtained as shown in Figure 2.

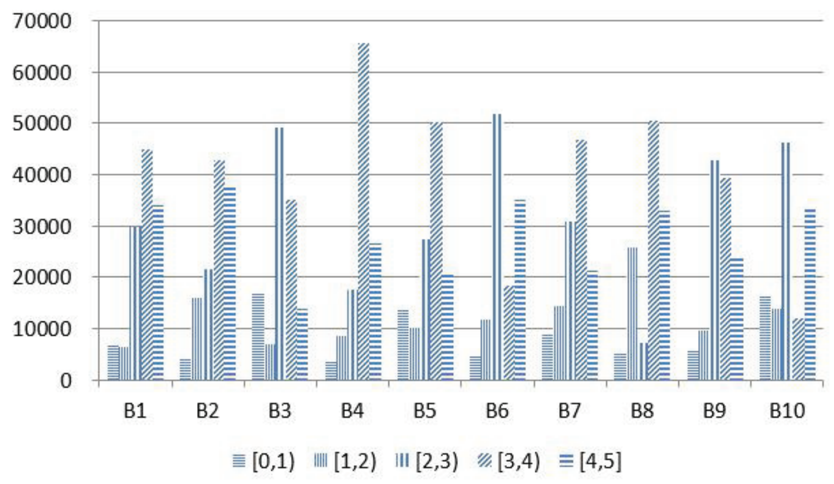

FIGURE 1. $s = 5\%$, block sampling data histogram

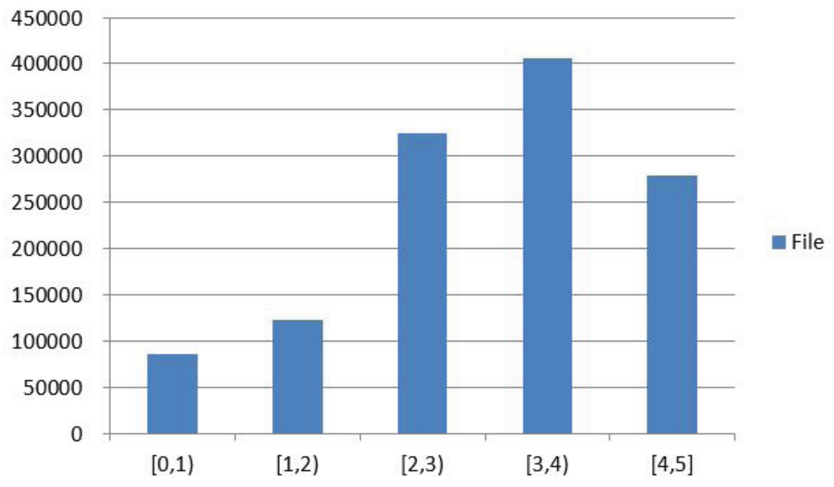

FIGURE 2. $s = 5\%$, file sampling data histogram

TABLE 5. Sampling data II

	$[0,1)$	$[1,2)$	$[2,3)$	$[3,4)$	$[4,5]$
Block 1	14692	37506	119532	100057	216295
Block 2	16410	47741	86670	211560	125701
Block 3	67255	27965	197016	140723	55123
Block 4	14636	34380	80516	232165	126385
Block 5	15262	40962	109852	240529	81477
Block 6	19000	47301	207693	113802	140286
Block 7	15479	17426	83228	257082	114867
Block 8	21474	63569	28651	182059	152329
Block 9	33822	58254	123154	185776	57076
Block 10	25771	55276	245103	58465	103467

When $s = 20\%$, the subgroup, the frequency of each group, and the statistics information in every Block are presented in Table 5.

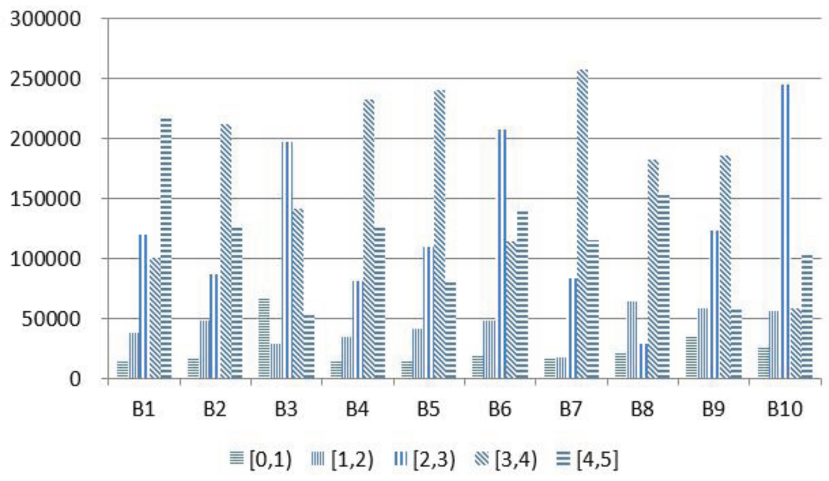

FIGURE 3. $s = 20\%$, block sampling data histogram

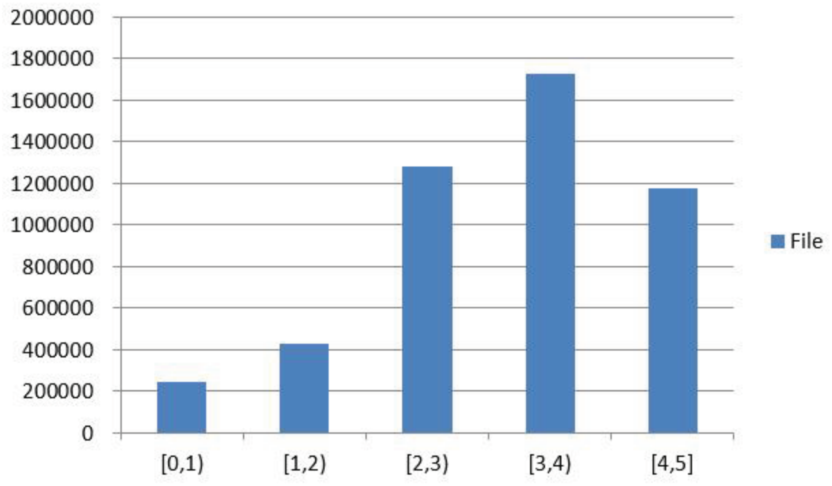

FIGURE 4. $s = 20\%$, file sampling data histogram

Use the sampling data information from the above table to construct the sampling data histogram of each Block which is presented in Figure 3.

All sampling data Block histogram information is combined on the Reduce node, and the histogram of the entire file sampling data is obtained as shown in Figure 4.

When $s = 100\%$, the subgroup, the frequency of each group and the statistics information in every Block are presented in Table 6.

The data histogram of every Block is constructed according to the above table as is shown in Figure 5.

All sampling data Block histogram information is combined on the Reduce node, and the histogram of the entire file sampling data is obtained as shown in Figure 6.

Through the contrast of Figure 1, Figure 3, and Figure 5, when the sampling coefficient $s = 5\%$, the difference between the sampling data in every Block and the real data is big, but when $s = 20\%$, the difference is relatively small. Through the contrast of Figure 2, Figure 4, and Figure 6, although the difference between the sampling data in every Block and the real data is obvious, the difference of the histogram of file sampling data and the histogram of the real data is relatively small, which can basically reflect the real file

TABLE 6. Sampling data III

	$[0, 1)$	$[1, 2)$	$[2, 3)$	$[3, 4)$	$[4, 5]$
Block 1	73456	187530	597664	500285	1081475
Block 2	82038	238708	433354	1057804	628506
Block 3	336277	139819	985084	703615	275615
Block 4	73184	171900	352580	1310825	531921
Block 4	76313	204812	549261	1202638	407386
Block 6	95002	236505	838461	569012	701430
Block 7	77397	87119	416143	1535412	324339
Block 8	107359	317847	143259	1210298	661647
Block 9	119114	191250	515772	1328890	285384
Block 10	128857	276382	1325504	42326	667337

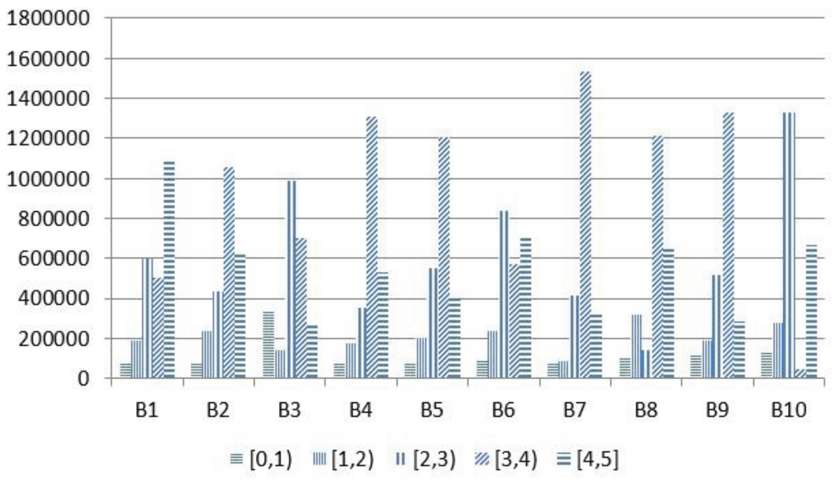

FIGURE 5. Block data histogram

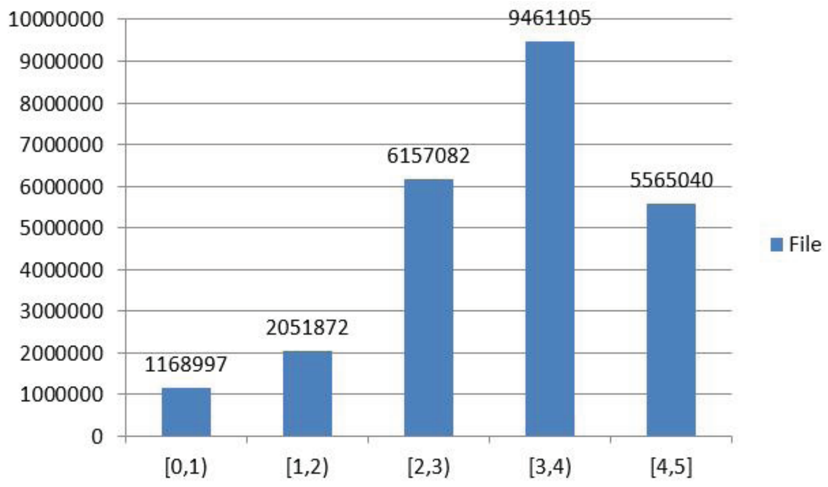

FIGURE 6. File data histogram

distribution status. Through many experiments, when the sampling coefficient reaches 10%-15%, the requirements can be met.

In addition, through the entire file data histogram, the data content itself in the entire file is not balanced distributed. The forth subgroup has obviously more data, the first subgroup has less data. Through the Block data histogram, the data distribution difference among several Blocks is obvious.

5.2.2. *Data balance algorithm.* 10 Blocks are distributed onto 4 storage nodes and the Block distribution status is supposed to be: three triple Block nodes and one single Block node. The two Block distribution algorithms are compared.

Plan 1: random distribution algorithm with file balancing deviation $fd = 2495267$.

The Block distribution status is presented in Table 7 and node data histogram is shown in Figure 7.

Table 7. Block distribution status I

	Node1	Node2	Node3	Node4
Block Numbers	3	3	3	1
Block Distribution	Blocks 1, 2, 3	Blocks 4, 5, 6	Blocks 7, 8, 9	Block 10

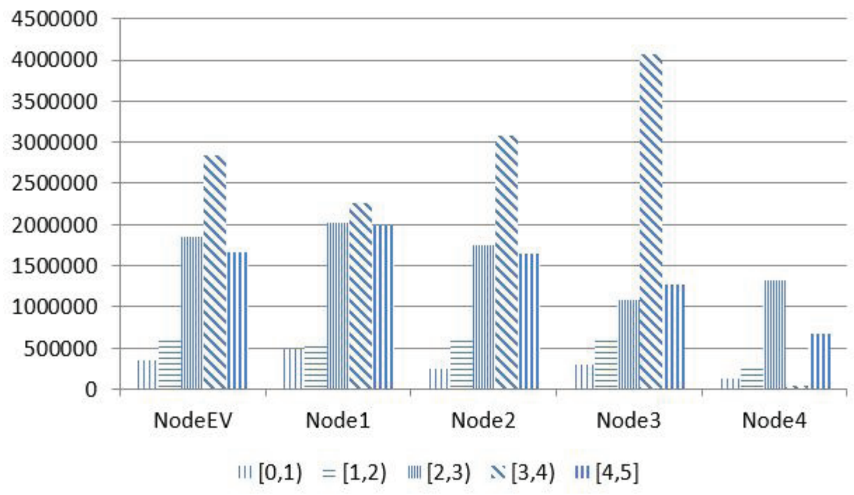

Figure 7. Node data histogram in the random algorithm

Plan 2: data balance algorithm with file balancing deviation $fd = 675231$. The Block distribution status is presented in Table 8 and node data histogram is shown in Figure 8.

Table 8. Block distribution status II

	Node1	Node2	Node3	Node4
Block Numbers	3	3	3	1
Block Distribution	Blocks 7, 8, 10	Blocks 2, 5, 6	Blocks 1, 3, 4	Block 9

NodeEV denotes the node expectation value in Figure 7 and Figure 8. Through the contrast among NodeEV and Node1-Node4, it can be seen that the histogram difference among NodeEV and Node1-Node4 is obvious in the random Block algorithm, and that the histogram difference among Node1 to Node4 is small in the data balance algorithm which

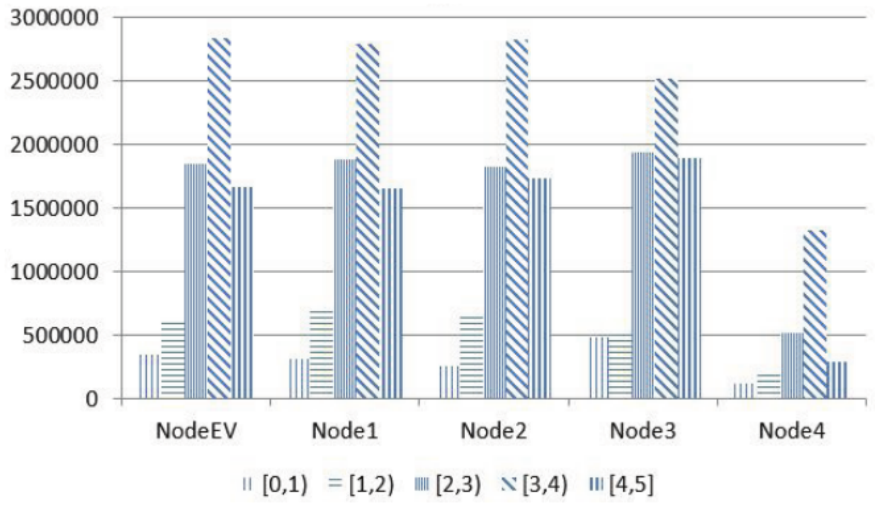

FIGURE 8. Node data histogram in the balance algorithm

is very close to the histogram distribution of NodeEV. Thus, the data balance algorithm has better data balancing effect than the random Block distribution algorithm. Through multiple experiment tests with different data, the data balance algorithm has significant improvement in the file balancing deviation compared with the random Block distribution algorithm (increased by $(2495267 - 675231)/2495267 = 72.94\%$ in the test result of this paper).

6. Conclusions.

Regarding the data skew in the Map side appearing in the MapReduce model, this paper proposes a data balance algorithm based on the sampling histogram. The proposed algorithm defines the file balancing deviation and reduces the file balancing deviation through exchanging the data blocks stored in the nodes under the condition of keeping the storing capacity of all nodes unchanged. Through multiple experiment tests with different data, the proposed algorithm has better balancing effect than the random Block distribution algorithm. It should be pointed out that the proposed algorithm uses the greedy strategy in the computing process and the strategy does not guarantee to find the global optimal solution to the data balancing distribution, but it can find a good approximate solution to the global optimal solution. Thus, the proposed method has great application value in the engineering field. In addition, the proposed algorithm combines three different research angles of data balance from the data content algorithm, data storage balance algorithm, and data computing balance algorithm. Such a multi-dimensional data balance study method, we believe, has potential research value and further research achievements are expected in the future.

Acknowledgment. This work is partially supported by the Sci.&Tec. Support Plan of Sichuan Province under the grant No. 2014GZ0013, the Sci.&Tec. Innovation Talent Project of Sichuan Province under the grant No. 2014-063, and the Scientific Research Project of CAFUC under the grant Nos. J2018-03 and F2017KF02. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which has improved the presentation.

REFERENCES

[1] Y. Yao et al., Self-adjusting slot configurations for homogeneous and heterogeneous Hadoop clusters, *IEEE Trans. Cloud Computing*, vol.5, no.2, pp.344-357, 2017.

[2] H. Won et al., Moving metadata from ad hoc files to database tables for robust, highly available, and scalable HDFS, *Journal of Supercomputing*, vol.73, no.6, pp.2657-2681, 2017.

[3] F. N. Afrati et al., Report from the third workshop on algorithms and systems for MapReduce and beyond, *SIGMOD Record*, vol.46, no.2, pp.43-48, 2017.

[4] G. Wang and S. Li, Research on handling data skew in MapReduce, *Computer Technology and Development*, vol.26, no.9, pp.201-204, 2016.

[5] V. A. Kostenko, Combinatorial optimization algorithms combining greedy strategies with a limited search procedure, *Journal of Computer and Systems Sciences International*, vol.56, no.2, pp.218-226, 2017.

[6] C. Zhang, F. Li and J. Jestes, Efficient parallel KNN joins for large data in MapReduce, *Proc. of the 15th International Conference on Extending Database Technology*, New York, pp.38-49, 2012.

[7] J. Jestes, K. Yi and F. Li, Building wavelet histograms on large data in MapReduce, *Proc. of the VLDB Endowment*, vol.5, no.2, pp.109-120, 2011.

[8] Y. Shi et al., HEDC++: An extended histogram estimator for data in the cloud, *Journal of Computer Science and Technology*, vol.28, no.6, pp.973-988, 2013.

[9] M. Tang, Efficient and scalable monitoring and summarization of large probabilistic data, *Proc. of the 2013 SIGMOD/PODS Ph.D. Symposium*, New York, pp.61-66, 2013.

[10] H. Singh and S. Bawa, A survey of traditional and MapReduce-based spatial query processing approaches, *SIGMOD Record*, vol.46, no.2, pp.18-29, 2017.

[11] Y. Kwon et al., A study of skew in MapReduce applications, *Open Cirrus Summit*, Moscow, Russia, 2011.

[12] J. Myungand et al., Handling data skew in join algorithms using MapReduce, *Expert Systems with Applications*, vol.51, pp.286-299, 2016.

[13] Y. He et al., MR-DBSCAN: A scalable MapReduce-based DBSCAN algorithm for heavily skewed data, *Frontiers of Computer Science*, vol.8, no.1, pp.83-99, 2014.

[14] Y. Cao, H. Lu, P. Duan, Y. Gao and L. Shi, Handling data skew in MapReduce programming model based on virtual partitioning method, *ICIC Express Letters*, vol.9, no.9, pp.2549-2553, 2015.

[15] B. Gufler et al., Load balancing in MapReduce based on scalable cardinality estimates, *Proc. of the 28th IEEE International Conference on Data Engineering (ICDE)*, Washington D.C., USA, pp.522-533, 2012.

[16] B. Gufler et al., Handing data skew in MapReduce, *Proc. of the 1st International Conference on Cloud Computing and Services Science*, Noordwijkerhout, Netherlands, vol.146, pp.574-583, 2011.

[17] L. Kolb, A. Thor and E. Rahm, Block-based load balancing for entity resolution with MapReduce, *Proc. of the 20th ACM International Conference on Information and Knowledge Management (CIKM)*, Glasgow, UK, pp.2397-2400, 2011.

[18] L. Kolb, A. Thor and E. Rahm, Load balancing for MapReduce-based entity resolution, *Proc. of the 28th IEEE International Conference on Data Engineering (ICDE)*, Washington D.C., USA, pp.618-629, 2012.

[19] Z. Wang et al., An incremental partitioning strategy for data balance on MapReduce, *Chinese Journal of Computers*, vol.1, pp.19-35, 2016.