ISBN 978-981-11-3671-9
Proceedings of 2017 the 7th International Workshop on Computer Science and Engineering (WCSE 2017)
Beijing, 25-27 June, 2017, pp.37-41
doi: 10.18178/wcse.2017.06.006

A Sampling Strategy for Skewed Data Problem in MapReduce

Cheng Wenjuan [1+], Tong Bing [2] , Zhou Miaomiao [3] and Zhu Junhong [4]

[1] School of Computer and Information, Hefei University of Technology
[2] School of Computer and Information, Hefei University of Technology
[3] School of Computer and Information, Hefei University of Technology
[4] School of Management, Hefei University of Technology

Abstract. As an efficient and reliable parallel computing model, MapReduce was widely used in all walks of life. However, when MapReduce dealing with the skewed data, the efficiency of the whole cluster will be reduced. And the load imbalance in reducer nodes will happen quite often after assigning the results from map stage. This paper used a reservoir sampling algorithm, it can sample with the same probability in case of unknown and skewed data set, thus we estimated the frequency of the Key in overall data, and then reallocated the tasks of processing node to achieve load balance. Finally, by comparing with the traditional sampling strategy, the experimental results showed that the method in this paper is more effective in case of computing skewed data set, advantages are more obvious with the increase of data set.

Keywords: MapReduce, Skewed data, Reservoir sampling, Load balancing

1. Introduction

With the development of Internet technology, the amount of information showing up at the pump, traditional data storage and processing cannot keep up with the growth of demand, data storage from the single node storage, disk array storage and other ways to the distributed storage, which makes the process of computing magnanimity data become an inevitable trend. MapReduce [1] was proposed by Google for the first time in 2004, and it has become mature increasingly since its birth, it is a kind of parallel computing model which is widely used at present [2], [3]. But when using MapReduce to process large data, generally, the distribution and the quantity of the data set are unknown, which will result in load imbalance in computing nodes and reducing its performance if data set were skewed, finally, the running time of the whole job is increased.

As is known to all, MapReduce can split the input file into many pieces, each file piece is processed by mapper function in parallel, and output a large number of key-value <key,value> pairs, then merge them through default Hash Partition function, so, the <key,list<value>> pairs are assigned to reducer processing node. When data sets are skewed, as shown in Fig. 1, the <key,list<value>> pairs from map stage are uneven distribution. In such situation, these skewed data in the calculation may lead to two problems:

1. The keys of skewed data cannot be evenly distributed to the reducer node through the default Hash function. For example, there is a data set D {a,b,c,d,e,f,g,h,i,j,k}, and three reducer nodes: reducer1, reducer2, reducer3, after Hash function, it will occur this situation: reducer1's load is {a,b,c,e,g,h,j,k}, reducer2's load is {d}, reducer3's load is {f,i}, then imbalance occurs. The basic reason is that the default hash function cannot balance the key to the reduce nodes.

+ Corresponding author.
E-mail address: *cheng@ah.edu.cn, ,eric_tongbing@163.com, miaomiaozhou@mail.hfut.edu.cn, hong263@263.net*

2. The values of each key are different, some keys with large number of values will be distributed to the reducer node, and the node will overload. As shown in Fig. 1, the key's value is uneven after Map stage, the tasks assigned to reducer2 node will be much more than the other two nodes, it means that load occurs imbalance, thus the running time of the reducer2 node will increase. Besides, the node with light load has finished the task already and come into idle state, and other nodes with overload are still computing, ultimately the cluster node utilization rate will decrease.

Fig.1: Schematic diagram of skewed data in MapReduce

In order to make each reducer node load balance, it is necessary to get the frequency distribution of Key to redefine a reasonable partition plan. We are unable to obtain the frequency of all key from the super large data set, but the sample method can be used effectively through the key frequency of the sample to calculate the key frequency of the total data.

2. Relate Work

Data Skew problem in MapReduce has been studied for a long time. When the scale of data became larger, the processing of the large data set becomes more complicated. However, the data skew problem can be solved by sampling to estimate all the key. YC Kwon sampled data and redefined a partition method to make load balance without discussing the size of sample[4]; Smriti R Ramakrishnan used a progressive sampling method to balance reducer skew in MapReduce workloads[5]; Geng Yujiao used a random sampling method, just selected the top k of the data as a sample to calculate the frequency of each Key, and provided a basis for data division[6],even if, it can solve the problem but randomness and blindness; So Li Haifeng proposed a two-stage sampling method to get a sample with representative data[7]; Grover R proposed an adaptive MapReduce sampling method to solve the problem[8].

3. Reservoir Sampling Algorithm

First of all, we consider the following question: How to extract k rows with equal probability randomly in a file, and the number of rows in file is unknown? Using reservoir sampling algorithm can be efficiently to solve it, the idea of algorithm as followings:

Step1: Selecting the first k lines as the sample;

Step2: Starting from the line k+1, with the probability of k/k+1 to decide whether to use the line k+1 to replace one of the k lines in the sample;

Step3: Deciding with the probability k/k+2 whether to use the line k+2 to replace one of the k lines in the sample;

Step4: Deciding with the probability k/k+3 whether to use the line k+3 to replace one of the k lines in the sample;

Step5: And so on, until the end of the file.

So another conclusion can be drawn from the above: In extending problem, the probability of each element as the sample is k/n.

Prove: Supposing an element i in a data stream of size n (n>=1), the probability Pi determines the element i as the sample, the probability Ci determines the element i to replace one of the k elements in the sample, the probability 1-1/k determines each element i in the sample not be replaced. That is:

$$\text{When } i <= k, Ci = 1 \text{ then } Pi = Ci \prod_{j=k+1}^{n} [(1 - \frac{k}{j}) + \frac{k}{j}(1 - \frac{1}{k})] = \frac{k}{n} \quad (1)$$

$$\text{When } i > k, Ci = \frac{k}{i} \text{ then } Pi = Ci \prod_{i=i+1}^{n} [(1 - Ci) + Ci(1 - \frac{1}{k})] = \frac{k}{n} \quad (2)$$

According to the above proofs, for each element of the overall data, the probability P=k/n can be used for sampling rate, the k is the capacity of the sample. It can be proved that the error generated by using p as the sampling rate is close to 0[9].

(1)Definition:Supposing a large data set $D\{key_1, key_2, key_3, \cdots key_n\}$, there are many equals k in data set D, we called it o(k), it represents the number of the key equals k in data set D. After sampling, s(k) represents the number of the key equals k in sample S.

(2)It's easy to get that the test times is o(k), s(k) exactly obeys the binomial distribution s(k)~B(o(k),p).

(3)In order to obtain the o(k) of the data set D, the unbiased estimate of o(k) is needed, and $\forall k \in D, \frac{s(k)}{p}$ is the unbiased estimation of o(k), is the standard deviation of o(k)

$$\sqrt{\frac{1-p}{p}o(k)} \text{ and } \sqrt{\frac{1-p}{p}o(k)} < \frac{n}{\sqrt{kk_1}} \tag{3}$$

Prove:

$$E(\hat{o}(k)) = E(\frac{s(k)}{p}) = \frac{1}{p}E(s(k)) = \frac{1}{p}o(k)p = o(k) \tag{4}$$

then $\frac{s(k)}{p}$ is unbiased estimation of o(k);

$$D(\hat{o}(k)) = D(\frac{s(k)}{p}) = \frac{1}{p^2}D(s(k)) = \frac{1}{p^2}o(k)p(1-p) = \frac{1-p}{p}o(k) \tag{5}$$

$\exists k_1 \in R$, supposing

$$o(k) = \frac{n}{k_1} \tag{6}$$

is the number of the key equals k in data set D, and

$$\frac{1-p}{p}o(k) < \frac{o(k)}{p} = \frac{n}{k}\frac{n}{k_1} = \frac{n^2}{kk_1} \tag{7}$$

then

$$\sqrt{\frac{1-p}{p}o(k)} < \frac{n}{\sqrt{kk_1}} \tag{8}$$

(4)According to the Moivre-Laplace central limit theorem, when n is big enough, binomial distribution approximately obeys the normal state distribution, then

$$\hat{o}(k) \sim N(o(k), \frac{1-p}{p}o(k)) \tag{9}$$

(5)According to 3σ theorem of the normal state distribution, we have

$$\forall k \in D,\ p\{|\hat{o}(k) - o(k)| \le 3\sigma\} = 2\Phi(3) - 1 \approx 99.74\% \tag{10}$$

$$p\{|\hat{o}(k) - o(k)| \le 3\frac{n}{\sqrt{kk_1}}\} > p\{|\hat{o}(k) - o(k)| \le 3\sigma\} \approx 99.74\% \tag{11}$$

The probability p is close to 1, and the bigger the value of k, the smaller the error. So it proves that the error generated by using p as the extraction probability is close to 0.

4. Sampling Module

MapReduce spends much more computing resources on processing large data, and it might add extra cost due to data skew problem, so a popular solution is pre-treated through sampling in Mapreduce to get the distribution of key frequency to repartition.

Reservoir sampling algorithm can randomly sample to get a representative sample from a large data set. The reservoir sampling algorithm is encapsulated in the initialization method of the ReservoirSampler-

RecordReader class. ReservoirSamplerRecordReader will correctly extract sample from the InputSplit that produced by the default InputFormat fragmentation mechanism in MapReduce, and it will input the sample file to mapper function to get the frequency of key, then according to the statistical result to repartition, Although it will spend some time on sampling data , the impact compared with the load imbalance caused by data skew is negligible. The sampling model is illustrated in Fig. 2.

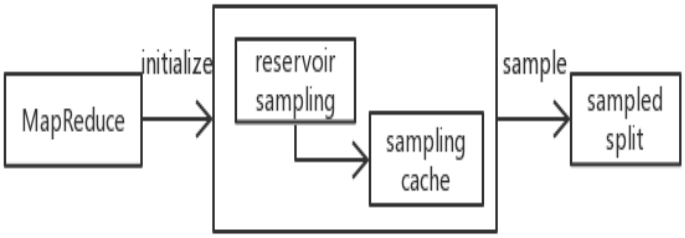

Fig. 2: Schematic diagram of reservoir sampling model

Algorithm pseudo code: Stream←Data Stream, Samples←a Sample with k elements

```
For ( int i = 0; i < k; i++)
        samples[i] = stream[i];
For (i = k; stream != null; i++) {
        r= random(1, i);
          if (r < k)
        samples[r] = stream[i];
    }
return samples;
```

5. Experiment and Result Analysis

In experiments, we built Hadoop cluster environment, there are 6 nodes in the cluster, one master node and five slave nodes, each node is configured with Intel Core I5, 2GB memory and 500GB hard disk. The version of Hadoop is hadoop-2.1.2, the operating system is Ubuntu-14.04.3,and system development environment for jdk-1.7.9.

We considered the audience's different loves for different popular songs, and this situation will cause the user’s behavioral data to have a certain inclination, so the data can meet our requirements, and user's behavior records from a music platform are used as test data in the experiments. The experiments compared the reservoir sampling with the traditional random sampling. At first, we selected 2GB data to compare the effect of sample’s size on the two methods. As shown in Fig. 3, when the sample’s size is relatively small, the two methods are very close to each other, because small volume sample is unrepresentative, reservoir sampling is similar to the traditional sampling. However, when the sample’s size increased, the total running time of both methods were reducing, and the gap between the two methods was obvious, the reservoir sampling method became much better than traditional sampling. Nevertheless, when the sample’s size was added to a certain degree, the sample time will increase, thus totaling running time will increase, but the reservoir sampling always better than the traditional sampling and maintaining the time advantage.

From the above analysis, we can see that the best sample size for 2GB data set is 1 million rows. Now, the influence of different data set on total running time is compared. The best effect of sample size for each data set had been selected. As shown in Fig. 4, when the data set were relatively small, the reservoir algorithm will consume more time, this is because compared to traditional methods, the sampling stage accounted for a greater proportion of the entire job. However, the advantage of reservoir sampling algorithm is increased by ever increasing volumes of test data, so that the total running time of entire job has plummeted. Therefore, the bigger of the amount of data, the more obvious advantage of the reservoir sampling algorithm.

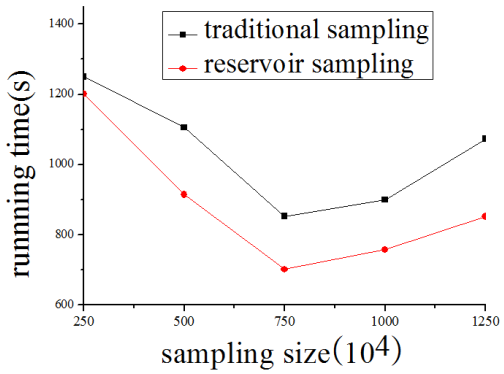

Fig. 3: Effect of sample size on running time

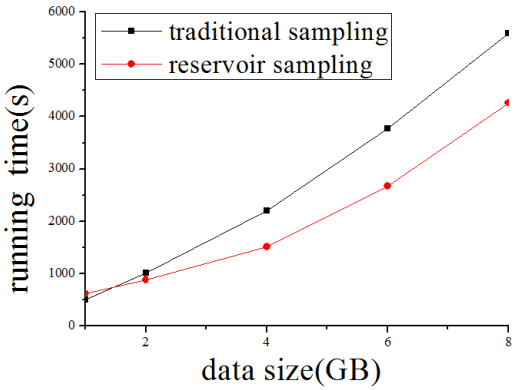

Fig. 4: Effect of data set size on running time

6. Conclusion

In this paper, we mainly studied the problem of load imbalance in reducer nodes, it is due to the skewed data in MapReduce. The reservoir sampling algorithm was used to obtain the frequency of key and redistribute the tasks with load balance to reduce nodes. It was analyzed theoretically and programmed in the Hadoop cluster system, then we selected the actual data set of the enterprise in the comparative experiments to analysis the reservoir sampling and the traditional sampling. The experiments proved that the reservoir sampling can handle the skewed data better than the traditional sampling. It makes the reducer nodes load balance and reduce the total running time.But the experimental environment is limited and the data used in the experiment are simple, so there is a certain gap with the actual working, what's more, the algorithm can only solve the problem in Reduce Stage, and the problem in Map stage will be considered soon.

7. Acknowledgment

This research was financially supported by the Quality Engineering of AnHui Province (2016ckjh141), and the Ministry of education of Humanities and Social Science project (JS2016JYRW0079), and the National Science Foundation (51274078).

8. References

[1] Dean J, Ghemawat S, "Mapreduce:Simplified Data Processing on Large Clusters," presented at the 6th Conference on Symposium on Operating Design&Implementation, 2004.

[2] T. Condie , "Mapreduce online," presented at the 7th USENIX Conference on Networked Systems Design and Implementation, 2010.

[3] L. K, P. J. Rousseeuw, "Finding groups in data: an introduction to cluster analysis," *John Wiley & Sons*, 2009.

[4] Kwon Y, "Skew-resistant Parallel Processing of Feature-extracting Scientific User-defined Functions," presented at the 1st ACM symposium on Cloud computing, 2010.

[5] Smriti R Ramakrishnan, "Balancing Reducer Skew in MapReduce Workloads using Progressive Sampling," presented at the ACM Symposium on Cloud Computing, 2012.

[6] Geng Yujiao, "The Research Skew with Sampling Technique in MapReduce,"M.A. Dissertation, Dept.Computer.Eng, Dalian Maritime University, 2013.

[7] Li Haifeng, "The Research of Load Balanceing in MapReduce Based on Sampling Estimation," M.A. Dissertation, Dept.Comput.Eng, Dalian Maritime University, 2014.

[8] Grover R,Carey M J, "Extending Map-reduce for Efficient Predicate—Based Sampling," presented at the 2012 IEEE 28th Int Conf on Data Engineering.Piscataway, 2012.

[9] Zou Peng, "Sampling-based Partitioning in MapReduce for Skewed Data," presented at the Seventh ChinaGrid Annual Conference, 2013.