Distributed log information Processing with Map-Reduce

A Case Study from Raw Data to Final Models

Mingyue Luo,Gang Liu
School of Electronic Engineering
Beijing University of Posts and Telecommunications
Beijing, China
E-mail: luomingyue0123@163.com

***Abstract*—With the high development of Internet, e-commerce websites now routinely have to work with log datasets which are up to a few terabytes in size. How to remove messy data timely with low cost and find out useful information is a problem we have to face. The mining process involves several steps from pre-processing the raw data to establishing the final models. In this paper we describe our method to solve the problem with Map-Reduce. Hadoop[7] is a Map-Reduce implementation develops open-source software for reliable, scalable, distributed computing. Several applications which we have proposed: data extracting, sum operation, join operation and clustering algorithm are applied on hadoop. We can apply them on data pre-processing and detect users with the same interests. In particular, we focus on clustering algorithms. A clustering algorithms which integrate SOM(Self-Organized Map) and fuzzy[13] logic is combined with Map-Reduce and we call it MRSF here. With the help of hadoop cluster, large calculation of jobs with MRSF can be accommodated easily by just adding more nodes or computers to the cluster. From the experiment, we show that MRSF can scale well and efficiently process and analyze extremely large datasets.**

Keywords-Distributed Data Mining, Map-Reduce, data pre-processing, join operation

I. INTRODUCTION

With the rapid development of Internet, e-commerce websites have brought unprecedented huge records from users. Behavior information of the web users are concealed in the web log. The web log mining can find characteristics and rules of the users' visiting behavior to improve the service quality to users. Clustering is one of technologies of data mining which applied by web log mining. Applying clustering to analysis users' visiting behavior can realize clustering of users according to their interest, thus it will help us to improve the web site's structure. We can also apply the information on recommend system. However, the information is covered in log files which are up to a few terabytes in size. Processing huge datasets have to consume large amount of computation. In general, distributed computing is a good solution. Computing tasks are assigned to multiple machines in parallel to improving processing speed.

SOM has been put forward in [2] by Kohonen in 1990. It is a type of artificial neural network that is trained using unsupervised learning. Generally, a SOM uses competitive, unsupervised training process to organize nodes to represent an input data set. The training algorithm is basically an on-line stochastic algorithm, which updates the values of the weight vectors at each step of the training process. However, the algorithm makes computational time to be increased exponentially in large-scale data domains [8]. Optimization or modification for the training process is needed for certain application. Parallelization has been one of the research directions for increasing the performance of the SOM [5][6]. We realize the parallelization of SOM with Map-Reduce.

In this paper, data pre-processing such as cleaning up useless information, identifying users and constructing Vector space model are solved with Map-Reduce framework. After above operations, we apply the processed data on clustering algorithms. We combine SOM and fuzzy logic with Map-Reduce framework. As MRSF is running on multiple nodes of hadoop in parallel, it can efficiently process and analyze extremely large datasets.

II. MAP-REDUCE AND ITS APPLICATIONS

Map-Reduce is a programming model and an associated implementation for processing and generating large data sets [1]. Fig.1 shows the data flow in Map-Reduce. We will introduce some applications of Map-Reduce framework in the following parts.

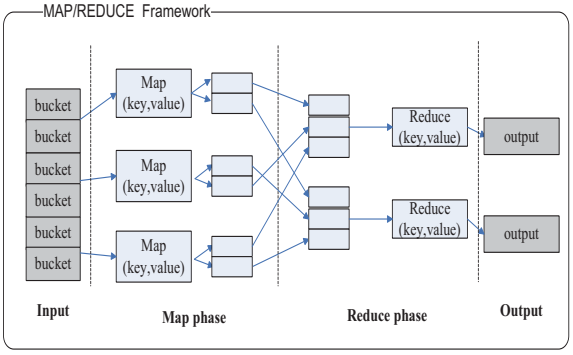

Figure1. Data flow of Map-Reudce

A. Simple data extracting and sum operation

In map stage it extracts useful information from raw records and transforms the information to an output key/value:

978-1-4244-6943-7/10/$26.00 ©2010 IEEE

Map (key1,value) -> list<key2,value2> .

In reduce stage it takes all values for a specific key, and generate a new list of the reduced output.

Reduce (key2, list<value2>) -> list<value3>

Usually, value3 is the values of sum operation in list<value2>.

B. Application to join operation

When we process datasets, we usually have to combine two files like join in SQL which is a query language in relational database. Join-operation often costs many resources of computation. If there are massive data and we do it on a single machine, it will waste a lot of time and space. Sometimes, join operation will not work out. However, With Map-Reduce framework, this can be solved.

- Add one column to each of two files which are prepared for join-operation. For example, the values of the two columns have constant values: tag1for file1 and tag2 for file2 in Table 1. File1 has three records and file2 has two records.

TABLE I. ADD TAGS TO FILES

	input	output
File1	a1 b1 c1 …	tag1 a1 b1 c1 …
	a1 b2 c2…	tag1 a1 b2 c2…
	a2 b3 c5…	tag1 a2 b3 c5…
File2	a1 d1 e1…	tag2 a1 d1 e1…
	a2 d1 e5…	tag2 a2 d1 e5…

- Read the two files at the same time. Each of records in files will be processed by map operation. Now the second columns which are to be join will become the output key of map and other column will be the values. We ensure that records which have the same key (second columns) can be partitioned into the same reduces. We can trace the change of the datasets in Table 2.

TABLE II. MAP OPERATION ON RECORDS

Map input	Map output
<rownum, tag1 a1 b1 c1 >	< a1, tag1 b1 c1 >
<rownum, tag1 a1 b2 c2 >	< a1, tag1 b2 c2 >
<rownum, tag1 a2 b3 c5>	< a2,tag1 b3 c5 >
<rownum, tag2 a1 d1 e1>	< a1, tag2 d1 e1>
<rownum, tag2 a2 d1 e5>	< a2, tag2 d1 e5>

- We make the output records of map stage to the input records of reduce stage. Records are partitions by hadoop and the records which have the same key are grouped into the same reducer. So in reduce input part there are only two records. In every record, each one in the list can be combined with others which have different tags. Then we can have the final result of join operation on the two file. We can clearly see it in table 3.

TABLE III. REDUCE OPERATION ON RECORDS

Reduce input	Reduce output
<a1, {tag1 b1 c1} {tag1 b2 c2} {tag2 d1 e1}>	a1 b1 c1 d1 e1 a 1 b2 c2 d1 e1
<a1, {tag1 b3 c5} {tag2 d1 e5}>	a1 b3 c5 d1 e5

C. Application to Clustering Algorithm

When an algorithm does sum over the data, we can easily distribute the calculations over hadoop cluster: We just divide the data set into as many pieces as maps, give each map its share of the data to sum the equations over, and aggregate the results at the end. We call this form of the algorithm the "summation form."[3]

1) Introduction to SOM with fuzzy logic

A SOM consists of two layers of nodes: an input layer and a competition (output) layer. The competition layer is actually an array with elements which is associated with an n-dimensional vector of weights, $M_j = (w_{1j}, w_{2j}, ..., w_{nj})$.

When an input vector x_i is presented to the output layer, the nodes in the output layer compete with each other. The winner will be the element M_j, which matches best with x_i.

The winning nodes are then updated according to the rule:

$$M_j(t+1) = M_j(t) + u_{ij} * (x_i - M_j(t)) \qquad (1)$$

where $M_j = (w_{1j}, w_{2j}, ..., w_{nj})$ is the center of $cluster_j$ and u_{ij} is a numerical value in [0, 1] that tells the degree to which the input element x_i belongs to the j-th cluster $M_j(t)$. As the closer the distance is, the greater the possibility belongs to $M_j(t-1)$. Thus, u_{ij} here is assigned in (2),

$$u_{ij}(t) = \frac{1}{\sum_{s=1}^{c} \left\{ \frac{\| x_i - M_j(t-1) \|^2}{\| x_i - M_s(t-1) \|^2} \right\}^{\frac{2}{K(t)-1}}} \qquad (2)$$

Where c is the number of categories and K(t) in (3) is a function which decreases over time.

$$K(t) = K_0 - t\left(\frac{K_0 - 1}{t_{\max}}\right) \qquad (3)$$

Where K_0 is a constant we set in advance and it must satisfy K_0>0. $t_{\max}$ is the maximum iteration. From (1) (2)

(3), we can get the adjustment of Cluster centers in (4):

$$\Delta M_j(t) = \sum_{i=1}^{n} \frac{(x_i - M_j(t-1))}{\sum_{s=1}^{c} \left\{ \frac{\| x_i - M_j(t-1) \|^2}{\| x_i - M_s(t-1) \|^2} \right\}} \quad (4)$$

Thus we can get the formula of $M_j(t)$

$$M_j(t) = M_j(t-1) + \Delta M_j(t) \quad (5)$$

2) MRSF clustering algorithm

It can be formalized as follows.

a) What we have to do in advance

- Initialize the number of cluster centers, c. Initialize $\Delta M_1(0)$, $\Delta M_2(0)$, … , $\Delta M_c(0)$ with very small value and $\sum_{j=1}^{c} \Delta M_j(t) < \varepsilon_2$ is satisfied . We write them into a file we call cluster-file0.
- Set the constants: $t_{\max}$, ε_1 , ε_2 and K_0. We must ensure $K_0 > 1$.

b) In map stage

In t-th iteration, we load the cluster-filej, and calculate $\sum_{j=1}^{c} \Delta M_j(t)$. If $\sum_{j=1}^{c} \Delta M_j(t) < \varepsilon_2$ is satisfied, we believe that we have found suitable cluster centers and stop the algorithm and set the output of maps null. Otherwise, For every input x_i, calculate u_{ij} (i=1,2,… n, j=1,2, … ,c) according to (2) which is the degree of the input vector x_i (i=1,2, … ,n) belonging to $M_j(t)$ (j=1,2, ..., c). If $u_{ij} > \varepsilon_1$, pick $M_j(t)$ as one of the winning cluster centers and set $u_{ij} = u_{ij}$. Otherwise, set $u_{ij} = 10^{-10}$. Then output the records in table 4.

TABLE IV. MAP OPERATION

Map input	Map output
< key, x_i >	< $M_j(t)$, x_i u_{ij} >

c) In reduce stage

In j-th reducer, after gaining pairs of <key,value> from maps in table 4, we calculate $\Delta M_j(t)$ in (4) and calculate M_{j+1} according to (5). Then we output $< M_{j+1}, \Delta M_j(t) >$ to one file on hadoop for the cluster. Table 5 shows the process.

TABLE V. REDUCE OPERATION

Reduce Input	Reduce Output
< $M_j(t)$, { x_1 u_{1j} } { x_2 u_{2j} }…{ x_n u_{nj} }>	< M_{j+1}, $\Delta M_j(t)$ >

III. IMPLEMENTATION

Visit-stream is a sequence of web pages the user access. It is composed of URLs and represents user's access behavior. Visit patterns are the behavior of users who visit a website. Through analyzing the visit behaviors of users, we can find their intention and interest.

Cookie logs of e-commerce website are our raw data. They are collected and pre-processed by us. Then we can apply MRSF algorithm to cluster the users. From Fig. 2, we can see the data processing procedures.

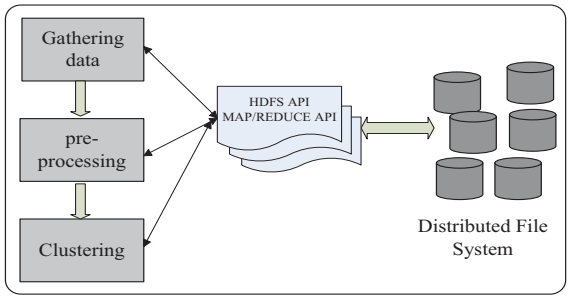

Figure2. Distributed mining process

A. Gathering data

We make several servers to collect the cookie logs simultaneously. These logs are transported to DFS (Distributed File System) by the scheduling system every day and wait for being processed.

B. Pre-processing

a) Getting rid of useless information

We make use of Map-Reduce to extract useful information in this section. Cookie ID, user ID and category ID of commodity which users click on are all obtained in map stage. In reduce stage we count the number of the same records. The result of sum operation is called CN (click number). In this way, we can get four columns: cookie ID, user ID, category ID and CN from the raw logs.

b) Filling the missing datasets

Cookie is a technology which is used in the original log for user identification. Many customers may Visit the website and do not login at first. However they may login later. If we only extract the information only with user ID, we may lose some useful information. We apply join operation we have explained in part II on the records to resolve the problem. We find a record which has both user ID and cookie ID, then fill user IDs with the record's user ID if the two records have same cookie ID. In this way the information of many records which have no user ID can be

obtained. After above processing, three columns: user ID, category ID and CN is held.

c) Making Vector Space Models(VSM)

We retain the top N category IDs which have most click numbers and make the top N category IDs as base vectors. Thus, CN becomes the value of category ID. In this way users' click behavior can be described as VSM which is shown in (6):

$$UC_i = (w_{i1}, w_{i2}, ..., w_{in}) \quad (6)$$

w_{ij} is the CN of category$_j$ which expresses the user$_i$' click number. Provided that the more click frequency in a category, the bigger w_{ij} is.

C. Data mining processing

Now we have had a vector model of users. We apply MRSF algorithm which is discussed above on it. Thus a final model is work out. We can get the clustering result in next part.

IV. EXPERIMENTAL RESULTS

We performed all experiments on our hadoop clusters which had different nodes. Computers in clusters had two dual-core processor with 4GB RAM and all running Linux. The size of raw data was about 50GB from a e-commerce websites. We could see the process time of pre-processing on hadoop cluster which have 15 nodes in table 6.

TABLE VI. THE TIME COST OF PER-PROCESS

	Input size	Output size	Cost time
extracting	50G	1.5G	1967s
Join operation	1.5G	1.5G	938s
Making VSM	1.5G	1.5G	469s

We focused on the experiments of clustering algorithm on hadoop. Speedup was made as the expression of the result. Speedup was defined by the following formula:

$$S_p = \frac{T_1}{T_p} \quad (7)$$

Where p was the number of processors; T_1 was the execution time of the sequential algorithm; T_p was the execution time of the parallel algorithm with p processors. We run MRSF on different nodes of hadoop cluster with the same parameters and got the result in Fig. 3. We could see that with the increase of nodes in hadoop, S_p was becoming bigger and the cost of time was becoming less.

ACKNOWLEDGMENT

In summary, this paper is a case study on data processing from raw log data to final model with Map-Reduce framework. With the help of hadoop cluster, huge data can be processed with shorter time but less cost. By just adding more nodes or computers to hadoop cluster, operations: Data extracting, sum operation, join operation and clustering algorithm can scale well and efficiently be processed.

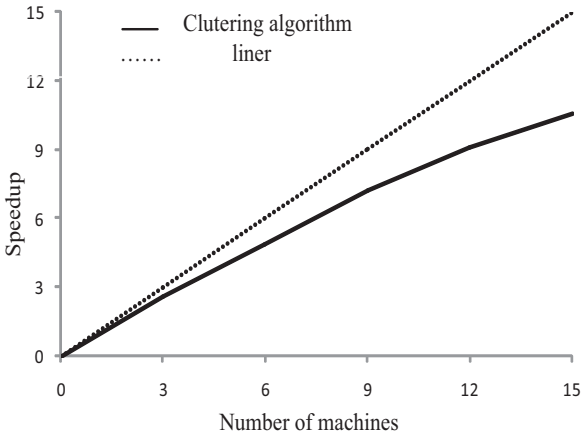

Figure3. The speedup of MRSF with different nodes in hadoop

REFERENCES

[1] J Dean, S Ghemawat," MapReduce: Simplified data processing on large clusters, " Communications of the ACM, 2008

[2] A. McCallum, K. Nigam and H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 169 - 178,2000

[3] Cheng-Tao Chu, Sang Kyun Kim and Yi-An Lin," Map-Reduce for Machine Learning on Multicore, " In NIPS , pp. 281-288, 2006

[4] Petri Vuorimaa, "Fuzzy self-organizing map, "Fuzzy Sets and Systems, pp 223-231, 1994.

[5] Eeagoe and A. Ropot, "Concurrent Self-Organizing Maps for Pattern Classification," Proceedings of the First IEEE International Conference of Cognitive Informatics (ICCI'02), 2002.

[6] Tianbai Qian and Minglu Li,"Multispectral MR Images Segmentation Using SOM Network, " Computer and Information Technology,pp.155-158, 2004.

[7] CK Fong,"A Study in Deploying Self-Organized Map(SOM) in an Open Source J2EE Cluster and Caching System, " 2007 IEEE/ICME International Conference on Complex Medical En2ineerin2, pp.778-781,2007.

[8] Mahout,http://mahout.apache.org/.

[9] S Papadimitriou, "DisCo: Distributed Co-clustering with Map-Reduce. A Case Study Towards Petabyte-Scale End-to-End Mining, " Data Mining ,ICDM '08. Eighth IEEE International Conference on, pp. 512 - 521 ,2008.

[10] T. Kohonen, "Self-Organizing Maps, " Springer-Verlag, Berlin, 3rd edition,2001.

[11] Tianyang Sun, Chengchun Shu," An Efficient Hierarchical Clustering Method for Large Datasets with Map-Reduce, " Parallel and Distributed Computing, Applications and Technologies, 2009 International Conference on, pp.494-495